Part I

Digital signatures

MURPHY LAW for CRYPTOGRAPHY

MURPHY LAW for CRYPTOGRAPHY

If there is a single security hole in a cryptosystem, the exposure of a cryptosystem will make sure that someone will eventually find it.

MURPHY LAW for CRYPTOGRAPHY

If there is a single security hole in a cryptosystem, the exposure of a cryptosystem will make sure that someone will eventually find it.

Even if this person is honest the discovery may ultimately leak to malicious parties.

MURPHY LAW for CRYPTOGRAPHY

If there is a single security hole in a cryptosystem, the exposure of a cryptosystem will make sure that someone will eventually find it.

Even if this person is honest the discovery may ultimately leak to malicious parties.

It is not suffcient that a cryptographic system is very secure, or even perfectly sucure - practically it is desirable that its implementations are secure enough what is vey hard to achieve.

CHAPTER 7: DIGITAL SIGNATURES

CHAPTER 7: DIGITAL SIGNATURES

Digital signatures are one of the most important inventions/applications of modern cryptography.

CHAPTER 7: DIGITAL SIGNATURES

Digital signatures are one of the most important inventions/applications of modern cryptography.

The problem is how can a user sign (electronically) an (electronic) message in such a way that everybody (or the intended addressee only) can verify the signature and signature should be good enough also for legal purposes.

CHAPTER 7: DIGITAL SIGNATURES

Digital signatures are one of the most important inventions/applications of modern cryptography.

The problem is how can a user sign (electronically) an (electronic) message in such a way that everybody (or the intended addressee only) can verify the signature and signature should be good enough also for legal purposes.

Moreover, a properly implemented digital signature should give the receiver a reason to believe that the received message was really send by the claimed sender (**authentication of the message**) and was not altered during the transit (**integrity of the message**).

Digital signatures are one of the most important inventions/applications of modern cryptography.

The problem is how can a user sign (electronically) an (electronic) message in such a way that everybody (or the intended addressee only) can verify the signature and signature should be good enough also for legal purposes.

Moreover, a properly implemented digital signature should give the receiver a reason to believe that the received message was really send by the claimed sender (**authentication of the message**) and was not altered during the transit (**integrity of the message**).

In many countries it is already desirable, or even necessay, to use in imporatnat communications digital signatures and they have also legal significance.

BASIC DEFINITION

Digital signature is a digital code (generated and authenticated by a public key encryption)

BASIC DEFINITION

Digital signature is a digital code (generated and authenticated by a public key encryption) which is attached to an electronically transmitted document

BASIC DEFINITION

Digital signature is a digital code (generated and authenticated by a public key encryption) which is attached to an electronically transmitted document to verify its contents and the sender's identity.

BASIC IDEAS

BASIC IDEAS

Example: Assume that each user A can use a special public-key cryptosystem (e_A, d_A).

BASIC IDEAS

Example: Assume that each user A can use a special public-key cryptosystem (e_A,d_A).

One way to sign a message w by a user A,

BASIC IDEAS

Example: Assume that each user A can use a special public-key cryptosystem (e_A,d_A).

One way to sign a message w by a user A, and to send w and its signature, so that any user can verify the signature,

BASIC IDEAS

Example: Assume that each user A can use a special public-key cryptosystem (e_A,d_A).

One way to sign a message w by a user A, and to send w and its signature, so that any user can verify the signature, is to apply on w (as the **signing procedure**) the mapping d_A :

BASIC IDEAS

Example: Assume that each user A can use a special public-key cryptosystem (e_A,d_A).

One way to sign a message w by a user A, and to send w and its signature, so that any user can verify the signature, is to apply on w (as the **signing procedure**) the mapping d_A :

sending message+signat.: $(w,\ \ d_A(w))) \rightarrow$

BASIC IDEAS

Example: Assume that each user A can use a special public-key cryptosystem (e_A,d_A).

One way to sign a message w by a user A, and to send w and its signature, so that any user can verify the signature, is to apply on w (as the **signing procedure**) the mapping d_A :

sending message+signat.: $(w, \; d_A(w))) \rightarrow$ **signature verification**:

BASIC IDEAS

Example: Assume that each user A can use a special public-key cryptosystem (e_A,d_A).

One way to sign a message w by a user A, and to send w and its signature, so that any user can verify the signature, is to apply on w (as the **signing procedure**) the mapping d_A :

sending message+signat.: $(w, \ d_A(w))) \rightarrow$ **signature verification**: $e_A(d_A(w)) = w$?

BASIC IDEAS

Example: Assume that each user A can use a special public-key cryptosystem (e_A,d_A).

One way to sign a message w by a user A, and to send w **and its signature, so that any user can verify the signature**, is to apply on w (as the **signing procedure**) the mapping d_A :

sending message+signat.: $(w,\ \ d_A(w)) \rightarrow$ **signature verification**: $e_A(d_A(w)) = w$?

One way to sign a message w by a user A so that only the user B can verify the signature,

BASIC IDEAS

Example: Assume that each user A can use a special public-key cryptosystem (e_A, d_A).

One way to sign a message w by a user A, and to send w and its signature, so that any user can verify the signature, is to apply on w (as the **signing procedure**) the mapping d_A :

sending message+signat.: $(w,\ d_A(w))) \rightarrow$ **signature verification**: $e_A(d_A(w)) = w$?

One way to sign a message w by a user A so that only the user B can verify the signature, is to apply on w (as the signing procedure) at first the mapping d_A and then, on the outcome, the mapping e_B:

sending message+signat.: $(w,\ e_B(d_A(w))) \rightarrow$

BASIC IDEAS

Example: Assume that each user A can use a special public-key cryptosystem (e_A, d_A).

One way to sign a message w by a user A, and to send w and its signature, so that any user can verify the signature, is to apply on w (as the **signing procedure**) the mapping d_A :

sending message+signat.: $(w,\ d_A(w))) \rightarrow$ **signature verification**: $e_A(d_A(w)) = w$?

One way to sign a message w by a user A so that only the user B can verify the signature, is to apply on w (as the signing procedure) at first the mapping d_A and then, on the outcome, the mapping e_B:

sending message+signat.: $(w,\ e_B(d_A(w))) \rightarrow$ **signature verification**:

BASIC IDEAS

Example: Assume that each user A can use a special public-key cryptosystem (e_A,d_A).

One way to sign a message w by a user A, and to send *w* and its signature, so that any user can verify the signature, is to apply on w (as the **signing procedure**) the mapping d_A :

sending message+signat.: $(w, \; d_A(w))) \rightarrow$ **signature verification**: $e_A(d_A(w)) = w$?

One way to sign a message w by a user A so that only the user B can verify the signature, is to apply on w (as the signing procedure) at first the mapping d_A and then, on the outcome, the mapping e_B:

sending message+signat.: $(w, \, e_B(d_A(w))) \rightarrow$ **signature verification**:
$e_A(d_B(e_B(d_A(w)))) = w$?

BASIC IDEAS

Example: Assume that each user A can use a special public-key cryptosystem (e_A,d_A).

One way to sign a message w by a user A, and to send *w* and its signature, so that any user can verify the signature, is to apply on w (as the **signing procedure**) the mapping d_A :

sending message+signat.: $(w, \ d_A(w)) \rightarrow$ **signature verification**: $e_A(d_A(w)) = w?$

One way to sign a message w by a user A so that only the user B can verify the signature, is to apply on w (as the signing procedure) at first the mapping d_A and then, on the outcome, the mapping e_B:

sending message+signat.: $(w, \ e_B(d_A(w))) \rightarrow$ **signature verification**:
$$e_A(d_B(e_B(d_A(w)))) = w?$$

A way to send a message w, and a signature of its hash, created by a user *A*, using a hash function h, so that anybody can verify the signature:

signing the hash: $(w, d_A(h(w))) \rightarrow$

BASIC IDEAS

Example: Assume that each user A can use a special public-key cryptosystem (e_A,d_A).

One way to sign a message w by a user A, and to send *w* and its signature, so that any user can verify the signature, is to apply on w (as the **signing procedure**) the mapping d_A :

sending message+signat.: $(w,\ \ d_A(w)) \rightarrow$ **signature verification**: $e_A(d_A(w)) = w?$

One way to sign a message w by a user A so that only the user B can verify the signature, is to apply on w (as the signing procedure) at first the mapping d_A and then, on the outcome, the mapping e_B:

sending message+signat.: $(w,\ e_B(d_A(w))) \rightarrow$ **signature verification**:
$e_A(d_B(e_B(d_A(w)))) = w?$

A way to send a message w, and a signature of its hash, created by a user A, using a hash function h, so that anybody can verify the signature:

signing the hash: $(w, d_A(h(w))) \rightarrow$ **signature verification**:

BASIC IDEAS

Example: Assume that each user A can use a special public-key cryptosystem (e_A,d_A).

One way to sign a message w by a user A, and to send w and its signature, so that any user can verify the signature, is to apply on w (as the **signing procedure**) the mapping d_A :

sending message+signat.: $(w,\ \ d_A(w)) \rightarrow$ **signature verification**: $e_A(d_A(w)) = w$?

One way to sign a message w by a user A so that only the user B can verify the signature, is to apply on w (as the signing procedure) at first the mapping d_A and then, on the outcome, the mapping e_B:

sending message+signat.: $(w,\ e_B(d_A(w))) \rightarrow$ **signature verification**:
$$e_A(d_B(e_B(d_A(w)))) = w?$$

A way to send a message w, and a signature of its hash, created by a user A, using a hash function h, so that anybody can verify the signature:

signing the hash: $(w, d_A(h(w))) \rightarrow$ **signature verification**: $h(w) = e_A(d_a(h(w)))$?

ADDITIONAL PROPERTIES of DIGITAL SIGNATURES

ADDITIONAL PROPERTIES of DIGITAL SIGNATURES

- In many ways and instances digital signatures provide a new layer of validation and security.

ADDITIONAL PROPERTIES of DIGITAL SIGNATURES

- In many ways and instances digital signatures provide a new layer of validation and security.
- **Digital signatures are both very different and also much equivalent to handwritten ones in many respects.**

- In many ways and instances digital signatures provide a new layer of validation and security.
- **Digital signatures are both very different and also much equivalent to handwritten ones in many respects.**

  Digital signatures, when properly implemented, are also more difficult to forge than handwritten signatures.

ADDITIONAL PROPERTIES of DIGITAL SIGNATURES

- In many ways and instances digital signatures provide a new layer of validation and security.
- **Digital signatures are both very different and also much equivalent to handwritten ones in many respects.**

  Digital signatures, when properly implemented, are also more difficult to forge than handwritten signatures.
- Digital signatures employ public-key cryptography.

DIGITAL SIGNATURES - OBSERVATION

Can we make digital signatures by digitizing our usual signature and attaching them to the messages (or documents) that need to be signed?

Can we make digital signatures by digitizing our usual signature and attaching them to the messages (or documents) that need to be signed?

No!

DIGITAL SIGNATURES - OBSERVATION

Can we make digital signatures by digitizing our usual signature and attaching them to the messages (or documents) that need to be signed?

No! Why?

Can we make digital signatures by digitizing our usual signature and attaching them to the messages (or documents) that need to be signed?

No! Why? Because such signatures could be easily removed and attached to some other documents or messages.

Can we make digital signatures by digitizing our usual signature and attaching them to the messages (or documents) that need to be signed?

No! Why? Because such signatures could be easily removed and attached to some other documents or messages.

Key observation: Digital signatures have to depend not only on the signer, but also on the document/message that is being signed.

DIGITAL SIGNATURES - BASIC REQUIREMENTS

DIGITAL SIGNATURES - BASIC REQUIREMENTS

Basic requirements - I. Digital signatures should be such that each user should be able to verify signatures of other users,

DIGITAL SIGNATURES - BASIC REQUIREMENTS

Basic requirements - I. Digital signatures should be such that each user should be able to verify signatures of other users, but that should give him/her no information how to sign a message on behalf of any other user.

DIGITAL SIGNATURES - BASIC REQUIREMENTS

Basic requirements - I. Digital signatures should be such that each user should be able to verify signatures of other users, but that should give him/her no information how to sign a message on behalf of any other user.

Basic requirements - II A valid digital signature should give the recipient reasons to believe that the message was created by a known sender and that it was not altered in transit.

DIGITAL SIGNATURES - BASIC REQUIREMENTS

Basic requirements - I. Digital signatures should be such that each user should be able to verify signatures of other users, but that should give him/her no information how to sign a message on behalf of any other user.

Basic requirements - II A valid digital signature should give the recipient reasons to believe that the message was created by a known sender and that it was not altered in transit.

Note An important difference from a handwritten signature is that digital signature of a message is always intimately connected with the message, and for different messages is different, whereas the **handwritten signature is adjoined to the message** and always looks the same.

DIGITAL SIGNATURES - BASIC REQUIREMENTS

Basic requirements - I. Digital signatures should be such that each user should be able to verify signatures of other users, but that should give him/her no information how to sign a message on behalf of any other user.

Basic requirements - II A valid digital signature should give the recipient reasons to believe that the message was created by a known sender and that it was not altered in transit.

Note An important difference from a handwritten signature is that digital signature of a message is always intimately connected with the message, and for different messages is different, whereas the **handwritten signature is adjoined to the message** and always looks the same.

Technically, a digital signature signing is performed by a **signing algorithm** and a digital signature is verified by a **verification algorithm**.

DIGITAL SIGNATURES - BASIC REQUIREMENTS

Basic requirements - I. Digital signatures should be such that each user should be able to verify signatures of other users, but that should give him/her no information how to sign a message on behalf of any other user.

Basic requirements - II A valid digital signature should give the recipient reasons to believe that the message was created by a known sender and that it was not altered in transit.

Note An important difference from a handwritten signature is that digital signature of a message is always intimately connected with the message, and for different messages is different, whereas the **handwritten signature is adjoined to the message** and always looks the same.

Technically, a digital signature signing is performed by a **signing algorithm** and a digital signature is verified by a **verification algorithm**.

A copy of a **digital (classical)** signature is identical (usually distinguishable) **to (from)** the origin. A care has therefore to be taken that digital signatures are not misused.

DIGITAL SIGNATURES - BASIC REQUIREMENTS

Basic requirements - I. Digital signatures should be such that each user should be able to verify signatures of other users, but that should give him/her no information how to sign a message on behalf of any other user.

Basic requirements - II A valid digital signature should give the recipient reasons to believe that the message was created by a known sender and that it was not altered in transit.

Note An important difference from a handwritten signature is that digital signature of a message is always intimately connected with the message, and for different messages is different, whereas the **handwritten signature is adjoined to the message** and always looks the same.

Technically, a digital signature signing is performed by a **signing algorithm** and a digital signature is verified by a **verification algorithm**.

A copy of a **digital (classical)** signature is identical (usually distinguishable) **to (from)** the origin. A care has therefore to be taken that digital signatures are not misused.

This chapter contains some of the main techniques for design and verification of digital signatures (as well as some possible attacks on them).

DIGITAL SIGNATURES - A PROBLEM

If only signature (but not the secrecy) of a message/document is of importance, then it suffices that Alice sends to Bob

$$(w, d_A(w))$$

DIGITAL SIGNATURES - A PROBLEM

If only signature (but not the secrecy) of a message/document is of importance, then it suffices that Alice sends to Bob

$$(w, d_A(w))$$

Caution: Signing a message w by A for B by

$$e_B(d_A(w))$$

is O.K., but the symmetric solution, with encoding first:

$$c = d_A(e_B(w))$$

is not good.

DIGITAL SIGNATURES - A PROBLEM

If only signature (but not the secrecy) of a message/document is of importance, then it suffices that Alice sends to Bob

$$(w, d_A(w))$$

Caution: Signing a message w by A for B by

$$e_B(d_A(w))$$

is O.K., but the symmetric solution, with encoding first:

$$c = d_A(e_B(w))$$

is not good.

Indeed, an active enemy, a tamperer T, can intercept the message, then can compute

$$d_T(e_A(c)) = d_T(e_B(w))$$

and can send the outcome to Bob,

DIGITAL SIGNATURES - A PROBLEM

If only signature (but not the secrecy) of a message/document is of importance, then it suffices that Alice sends to Bob

$$(w, d_A(w))$$

Caution: Signing a message w by A for B by

$$e_B(d_A(w))$$

is O.K., but the symmetric solution, with encoding first:

$$c = d_A(e_B(w))$$

is not good.

Indeed, an active enemy, a tamperer T, can intercept the message, then can compute

$$d_T(e_A(c)) = d_T(e_B(w))$$

and can send the outcome to Bob, pretending that it is from him/tamperer

DIGITAL SIGNATURES - A PROBLEM

If only signature (but not the secrecy) of a message/document is of importance, then it suffices that Alice sends to Bob

$$(w, d_A(w))$$

Caution: Signing a message w by A for B by

$$e_B(d_A(w))$$

is O.K., but the symmetric solution, with encoding first:

$$c = d_A(e_B(w))$$

is not good.

Indeed, an active enemy, a tamperer T, can intercept the message, then can compute

$$d_T(e_A(c)) = d_T(e_B(w))$$

and can send the outcome to Bob, pretending that it is from him/tamperer (without being able to decrypt/know the message).

DIGITAL SIGNATURES - A PROBLEM

If only signature (but not the secrecy) of a message/document is of importance, then it suffices that Alice sends to Bob

$$(w, d_A(w))$$

Caution: Signing a message w by A for B by

$$e_B(d_A(w))$$

is O.K., but the symmetric solution, with encoding first:

$$c = d_A(e_B(w))$$

is not good.

Indeed, an active enemy, a tamperer T, can intercept the message, then can compute

$$d_T(e_A(c)) = d_T(e_B(w))$$

and can send the outcome to Bob, pretending that it is from him/tamperer (without being able to decrypt/know the message).

Any public-key cryptosystem in which the plaintext and cryptotext spaces are the same can be used for digital signature.

WHY TO SIGN HASHES of MESSAGES and not MESSAGES THEMSELVES

Signing hashes of messages -example:

WHY TO SIGN HASHES of MESSAGES and not MESSAGES THEMSELVES

Signing hashes of messages -example:

A way to send a message w, and a signature of its hash, created by a user A, using a hash function h, so that any one can verify the signature:

signing the hash: $(w, d_A(h(w)))$

WHY TO SIGN HASHES of MESSAGES and not MESSAGES THEMSELVES

Signing hashes of messages -example:

A way to send a message w, and a signature of its hash, created by a user A, using a hash function h, so that any one can verify the signature:

signing the hash: $(w, d_A(h(w)))$ **signature verification**:

WHY TO SIGN HASHES of MESSAGES and not MESSAGES THEMSELVES

Signing hashes of messages -example:

A way to send a message w, and a signature of its hash, created by a user A, using a hash function h, so that any one can verify the signature:

signing the hash: $(w, d_A(h(w)))$ **signature verification**: $h(w) = e_A(d_a(h(w)))$

WHY TO SIGN HASHES of MESSAGES and not MESSAGES THEMSELVES

Signing hashes of messages -example:

A way to send a message w, and a signature of its hash, created by a user A, using a hash function h, so that any one can verify the signature:

signing the hash: $(w, d_A(h(w)))$ **signature verification**: $h(w) = e_A(d_a(h(w)))$

There are several reasons why it is better to sign hashes of messages than messages themselves.

WHY TO SIGN HASHES of MESSAGES and not MESSAGES THEMSELVES

Signing hashes of messages -example:

A way to send a message w, and a signature of its hash, created by a user A, using a hash function h, so that any one can verify the signature:

signing the hash: $(w, d_A(h(w)))$ **signature verification**: $h(w) = e_A(d_a(h(w)))$

There are several reasons why it is better to sign hashes of messages than messages themselves.

- **For efficiency**: Hashes are much shorter and so are their signatures - this is a way to save resources (time,...)

WHY TO SIGN HASHES of MESSAGES and not MESSAGES THEMSELVES

Signing hashes of messages -example:

A way to send a message w, and a signature of its hash, created by a user A, using a hash function h, so that any one can verify the signature:

signing the hash: $(w, d_A(h(w)))$ **signature verification**: $h(w) = e_A(d_a(h(w)))$

There are several reasons why it is better to sign hashes of messages than messages themselves.

- **For efficiency**: Hashes are much shorter and so are their signatures - this is a way to save resources (time,...)
- **For compatibility:** Messages are typically bit strings. Digital signature schemes, such as RSA, operate often on other domains. A hash function can be used to convert an arbitrary input into the proper form.

WHY TO SIGN HASHES of MESSAGES and not MESSAGES THEMSELVES

Signing hashes of messages -example:

A way to send a message w, and a signature of its hash, created by a user A, using a hash function h, so that any one can verify the signature:

signing the hash: $(w, d_A(h(w)))$ **signature verification**: $h(w) = e_A(d_a(h(w)))$

There are several reasons why it is better to sign hashes of messages than messages themselves.

- **For efficiency**: Hashes are much shorter and so are their signatures - this is a way to save resources (time,...)
- **For compatibility:** Messages are typically bit strings. Digital signature schemes, such as RSA, operate often on other domains. A hash function can be used to convert an arbitrary input into the proper form.
- **For integrity:** If hashing is not used, a message has to be often split into blocks and each block signed separately.

WHY TO SIGN HASHES of MESSAGES and not MESSAGES THEMSELVES

Signing hashes of messages -example:

A way to send a message w, and a signature of its hash, created by a user A, using a hash function h, so that any one can verify the signature:

signing the hash: $(w, d_A(h(w)))$ **signature verification**: $h(w) = e_A(d_a(h(w)))$

There are several reasons why it is better to sign hashes of messages than messages themselves.

- **For efficiency**: Hashes are much shorter and so are their signatures - this is a way to save resources (time,...)
- **For compatibility:** Messages are typically bit strings. Digital signature schemes, such as RSA, operate often on other domains. A hash function can be used to convert an arbitrary input into the proper form.
- **For integrity:** If hashing is not used, a message has to be often split into blocks and each block signed separately. However, the receiver may not able to find out whether all blocks have been signed and sent in the proper order.

A GENERAL SCHEME of DIGITAL SIGNATURE SYSTEMS – SIMPLIFIED VERSION

A GENERAL SCHEME of DIGITAL SIGNATURE SYSTEMS – SIMPLIFIED VERSION

A **digital signature system** (**DSS**) consists of:

A GENERAL SCHEME of DIGITAL SIGNATURE SYSTEMS – SIMPLIFIED VERSION

A **digital signature system** (**DSS**) consists of:

- **P** - the space of possible plaintexts (messages/documents).

A GENERAL SCHEME of DIGITAL SIGNATURE SYSTEMS – SIMPLIFIED VERSION

A **digital signature system** (**DSS**) consists of:

- **P** - the space of possible plaintexts (messages/documents).
- **S** - the space of possible signatures.
- **K** - the space of possible keys.

A GENERAL SCHEME of DIGITAL SIGNATURE SYSTEMS – SIMPLIFIED VERSION

A **digital signature system** (**DSS**) consists of:

- **P** - the space of possible plaintexts (messages/documents).
- **S** - the space of possible signatures.
- **K** - the space of possible keys.
- For each $k \in K$ there is a **signing algorithm** sig_k and a corresponding **verification algorithm** ver_k such that

$$sig_k : P \to S.$$

$$ver_k : P \otimes S \to \{true, false\}$$

and

A GENERAL SCHEME of DIGITAL SIGNATURE SYSTEMS – SIMPLIFIED VERSION

A **digital signature system** (**DSS**) consists of:

- **P** - the space of possible plaintexts (messages/documents).
- **S** - the space of possible signatures.
- **K** - the space of possible keys.
- For each $k \in K$ there is a **signing algorithm** sig_k and a corresponding **verification algorithm** ver_k such that

$$sig_k : P \rightarrow S.$$

$$ver_k : P \otimes S \rightarrow \{true, false\}$$

and

$$\mathbf{ver}_k(w, s) = \begin{cases} true & \text{if } \; s = \mathbf{sig}_k(w); , \\ false & \text{otherwise.} \end{cases}$$

A GENERAL SCHEME of DIGITAL SIGNATURE SYSTEMS – SIMPLIFIED VERSION

A **digital signature system** (**DSS**) consists of:

- **P** - the space of possible plaintexts (messages/documents).
- **S** - the space of possible signatures.
- **K** - the space of possible keys.
- For each $k \in K$ there is a **signing algorithm** sig_k and a corresponding **verification algorithm** ver_k such that

  $$sig_k : P \to S.$$

  $$ver_k : P \otimes S \to \{true, false\}$$

  and

  $$\mathbf{ver}_k(w, s) = \begin{cases} true & \text{if } \ s = \mathbf{sig}_k(w);, \\ false & \text{otherwise.} \end{cases}$$

Algorithms sig_k and ver_k should be realizable in polynomial time.

A GENERAL SCHEME of DIGITAL SIGNATURE SYSTEMS – SIMPLIFIED VERSION

A **digital signature system** (**DSS**) consists of:

- **P** - the space of possible plaintexts (messages/documents).
- **S** - the space of possible signatures.
- **K** - the space of possible keys.
- For each $k \in K$ there is a **signing algorithm** sig_k and a corresponding **verification algorithm** ver_k such that

$$sig_k : P \to S.$$

$$ver_k : P \otimes S \to \{true, false\}$$

and

$$\mathbf{ver}_k(w,s) = \begin{cases} true & \text{if } \ s = \mathbf{sig}_k(w);, \\ false & \text{otherwise.} \end{cases}$$

Algorithms sig_k and ver_k should be realizable in polynomial time.

Verification algorithms can be publicly known; signing algorithms (actually only their keys) should be kept secret

DIGITAL SIGNATURE SCHEMES I

DIGITAL SIGNATURE SCHEMES I

Digital signature schemes are basic tools for authentication messages. A digital signature scheme allows anyone to verify signature of any sender S without providing any information how to generate signatures of S.

DIGITAL SIGNATURE SCHEMES I

Digital signature schemes are basic tools for authentication messages. A digital signature scheme allows anyone to verify signature of any sender S without providing any information how to generate signatures of S.

DIGITAL SIGNATURE SCHEMES I

Digital signature schemes are basic tools for authentication messages. A digital signature scheme allows anyone to verify signature of any sender S without providing any information how to generate signatures of S.

A **Digital Signature Scheme** (M, S, K_s, K_v) is given by:

DIGITAL SIGNATURE SCHEMES I

Digital signature schemes are basic tools for authentication messages. A digital signature scheme allows anyone to verify signature of any sender S without providing any information how to generate signatures of S.

A **Digital Signature Scheme** (M, S, K_s, K_v) is given by:

- M - a set of **messages** to be signed

DIGITAL SIGNATURE SCHEMES I

Digital signature schemes are basic tools for authentication messages. A digital signature scheme allows anyone to verify signature of any sender S without providing any information how to generate signatures of S.

A **Digital Signature Scheme** (M, S, K_s, K_v) is given by:

- M - a set of **messages** to be signed
- S - a set of possible **signatures**

DIGITAL SIGNATURE SCHEMES I

Digital signature schemes are basic tools for authentication messages. A digital signature scheme allows anyone to verify signature of any sender S without providing any information how to generate signatures of S.

A **Digital Signature Scheme** (M, S, K_s, K_v) is given by:

- M - a set of **messages** to be signed
- S - a set of possible **signatures**
- K_s - a set of **private keys for signing** - one for each signer

DIGITAL SIGNATURE SCHEMES I

Digital signature schemes are basic tools for authentication messages. A digital signature scheme allows anyone to verify signature of any sender S without providing any information how to generate signatures of S.

A **Digital Signature Scheme** (M, S, K_s, K_v) is given by:

- M - a set of **messages** to be signed
- S - a set of possible **signatures**
- K_s - a set of **private keys for signing** - one for each signer
- K_v - a set of **public keys for verification** - one for each signer

DIGITAL SIGNATURE SCHEMES I

Digital signature schemes are basic tools for authentication messages. A digital signature scheme allows anyone to verify signature of any sender S without providing any information how to generate signatures of S.

A **Digital Signature Scheme** (M, S, K_s, K_v) is given by:

- M - a set of **messages** to be signed
- S - a set of possible **signatures**
- K_s - a set of **private keys for signing** - one for each signer
- K_v - a set of **public keys for verification** - one for each signer

Moreover, it is required that:

- For each k from K_s, there exists a single and easy to compute **signing mapping**

$$sig_k\colon \{0,1\}^* \times M \to S$$

DIGITAL SIGNATURE SCHEMES I

Digital signature schemes are basic tools for authentication messages. A digital signature scheme allows anyone to verify signature of any sender S without providing any information how to generate signatures of S.

A **Digital Signature Scheme** (M, S, K_s, K_v) is given by:

- M - a set of **messages** to be signed
- S - a set of possible **signatures**
- K_s - a set of **private keys for signing** - one for each signer
- K_v - a set of **public keys for verification** - one for each signer

Moreover, it is required that:

- For each k from K_s, there exists a single and easy to compute **signing mapping**

$$sig_k\colon \{0,1\}^* \times M \to S$$

- For each k from K_v there exists a single and easy to compute **verification mapping**

$$ver_k\colon M \times S \to \{true,\ false\}$$

such that the following two conditions are satisfied:

DIGITAL SIGNATURES SCHEMES II - conditions

Correctness:

DIGITAL SIGNATURES SCHEMES II - conditions

Correctness:

For each message m from M and public key k from K_v, it should hold

$$ver_k(m, s) = \text{true}$$

if there is an r from $\{0,1\}^*$ such that

$$s = sig_l(r, m)$$

for a private key l from K_s corresponding to the public key k.

DIGITAL SIGNATURES SCHEMES II - conditions

Correctness:

For each message m from M and public key k from K_v, it should hold

$$ver_k(m, s) = \text{true}$$

if there is an r from $\{0, 1\}^*$ such that

$$s = sig_l(\text{r, m})$$

for a private key l from K_s corresponding to the public key k.

Security:

DIGITAL SIGNATURES SCHEMES II - conditions

Correctness:

For each message m from M and public key k from K_v, it should hold

$$ver_k(m, s) = \text{true}$$

if there is an r from $\{0, 1\}^*$ such that

$$\text{s} = sig_l(\text{r, m})$$

for a private key l from K_s corresponding to the public key k.

Security:

For any w from M and k from K_v, it should be computationally unfeasible, without the knowledge of the private key corresponding to k, to find a signature s from S such that

$$ver_k(\text{w, s}) = \text{true}.$$

A COMMENT ON DIGITAL SIGNATURE SCHEMES

A COMMENT ON DIGITAL SIGNATURE SCHEMES

Sometimes it is required that a digital signature scheme contains also a **keys generation phase**, Sometimes it is required that a digital signature scheme contains also a **keys generation phase**,

It is a phase that creates uniformly and randomly a secret (signing) key (from a set of potential secret keys) and outputs this secret key and the corresponding public (verification) key.

ADDITIONAL PROPERTIES Of DIGITAL SIGNATURES

ADDITIONAL PROPERTIES Of DIGITAL SIGNATURES

- Digital signatures can also provide so-called **non-repudiation**.

- Digital signatures can also provide so-called **non-repudiation**. That means that the signer cannot successfully claim that he did not signed a message, while also claiming that his private key remains secret.

ADDITIONAL PROPERTIES Of DIGITAL SIGNATURES

- Digital signatures can also provide so-called **non-repudiation**. That means that the signer cannot successfully claim that he did not signed a message, while also claiming that his private key remains secret.

- **An encryption system is considered as broken if one can determine (at least a part of) plaintexts from at least some cryptotexts (and at least sometimes).**

BREAKING DIGITAL SIGNATURE SYSTEMS

- **An encryption system is considered as broken if one can determine (at least a part of) plaintexts from at least some cryptotexts (and at least sometimes).**
- **A digital signature system is considered as broken if one can (at least sometimes) forge (at least some) signatures.**

- **An encryption system is considered as broken if one can determine (at least a part of) plaintexts from at least some cryptotexts (and at least sometimes).**
- **A digital signature system is considered as broken if one can (at least sometimes) forge (at least some) signatures.**
- **In both cases, a more ambitious goal is to find the private key.**

ATTACKS MODELS on DIGITAL SIGNATURES

ATTACKS MODELS on DIGITAL SIGNATURES

Basic attack models

ATTACKS MODELS on DIGITAL SIGNATURES

Basic attack models

KEY-ONLY ATTACK: The attacker is only given the public verification key.

ATTACKS MODELS on DIGITAL SIGNATURES

Basic attack models

KEY-ONLY ATTACK: The attacker is only given the public verification key.

KNOWN SIGNATURES ATTACK: The attacker is given valid signatures for several messages known, but not chosen, by the attacker.

ATTACKS MODELS on DIGITAL SIGNATURES

Basic attack models

KEY-ONLY ATTACK: The attacker is only given the public verification key.

KNOWN SIGNATURES ATTACK: The attacker is given valid signatures for several messages known, but not chosen, by the attacker.

CHOSEN SIGNATURES ATTACK: The attacker is given valid signatures for several messages chosen by the attacker.

ATTACKS MODELS on DIGITAL SIGNATURES

Basic attack models

KEY-ONLY ATTACK: The attacker is only given the public verification key.

KNOWN SIGNATURES ATTACK: The attacker is given valid signatures for several messages known, but not chosen, by the attacker.

CHOSEN SIGNATURES ATTACK: The attacker is given valid signatures for several messages chosen by the attacker.

ADAPTIVE CHOSEN SIGNATURES ATTACKS: The attacker is given valid signatures for several messages chosen by the attacker where messages chosen may depend on previous signatures given for chosen messages.

LEVELS of BREAKING of DIGITAL SIGNATURES

- **Total break** of a signature scheme: The adversary manages to recover the secret key from the public key.

LEVELS of BREAKING of DIGITAL SIGNATURES

- **Total break** of a signature scheme: The adversary manages to recover the secret key from the public key.
- **Universal forgery:** The adversary can derive from the public key an algorithm which allows to forge the signature of any message.

LEVELS of BREAKING of DIGITAL SIGNATURES

- **Total break** of a signature scheme: The adversary manages to recover the secret key from the public key.
- **Universal forgery:** The adversary can derive from the public key an algorithm which allows to forge the signature of any message.
- **Selective forgery:** The adversary can derive from the public key a method to forge signatures of selected messages (where selection was made a priory the knowledge of the public key).

LEVELS of BREAKING of DIGITAL SIGNATURES

- **Total break** of a signature scheme: The adversary manages to recover the secret key from the public key.
- **Universal forgery:** The adversary can derive from the public key an algorithm which allows to forge the signature of any message.
- **Selective forgery:** The adversary can derive from the public key a method to forge signatures of selected messages (where selection was made a priory the knowledge of the public key).
- **Existential forgery:** The adversary is able to create from the public key a valid signature of a message m (but has no control for which m).

LEVELS of BREAKING of DIGITAL SIGNATURES

- **Total break** of a signature scheme: The adversary manages to recover the secret key from the public key.
- **Universal forgery:** The adversary can derive from the public key an algorithm which allows to forge the signature of any message.
- **Selective forgery:** The adversary can derive from the public key a method to forge signatures of selected messages (where selection was made a priory the knowledge of the public key).
- **Existential forgery:** The adversary is able to create from the public key a valid signature of a message m (but has no control for which m).

Observe that to forge a signature scheme means to produce a new signature - it is not forgery to obtain from the signer a valid signature.

A DIGITAL SIGNATURE of one BIT

A DIGITAL SIGNATURE of one BIT

Let us start with a very simple, but much illustrative (though non-practical), example how to sign a single bit.

A DIGITAL SIGNATURE of one BIT

Let us start with a very simple, but much illustrative (though non-practical), example how to sign a single bit.

Design of the signature scheme:

A DIGITAL SIGNATURE of one BIT

Let us start with a very simple, but much illustrative (though non-practical), example how to sign a single bit.

Design of the signature scheme:

A one-way function f(x) is publicly chosen.

A DIGITAL SIGNATURE of one BIT

Let us start with a very simple, but much illustrative (though non-practical), example how to sign a single bit.

Design of the signature scheme:

A one-way function f(x) is publicly chosen.

Two integers k_0 and k_1 are chosen and kept **secret** by the signer.

A DIGITAL SIGNATURE of one BIT

Let us start with a very simple, but much illustrative (though non-practical), example how to sign a single bit.

Design of the signature scheme:

A one-way function f(x) is publicly chosen.

Two integers k_0 and k_1 are chosen and kept **secret** by the signer. Three items

$$f,\ (0,\ s_0),\ (1,\ s_1)$$

are made public, where

$$s_0 = f(k_0),\ s_1 = f(k_1)$$

A DIGITAL SIGNATURE of one BIT

Let us start with a very simple, but much illustrative (though non-practical), example how to sign a single bit.

Design of the signature scheme:

A one-way function f(x) is publicly chosen.

Two integers k_0 and k_1 are chosen and kept **secret** by the signer. Three items

$$f, (0, s_0), (1, s_1)$$

are made public, where

$$s_0 = f(k_0), \; s_1 = f(k_1)$$

Signature of a bit b:

A DIGITAL SIGNATURE of one BIT

Let us start with a very simple, but much illustrative (though non-practical), example how to sign a single bit.

Design of the signature scheme:

A one-way function f(x) is publicly chosen.

Two integers k_0 and k_1 are chosen and kept **secret** by the signer. Three items

$$f, (0, s_0), (1, s_1)$$

are made public, where

$$s_0 = f(k_0), s_1 = f(k_1)$$

Signature of a bit b:

$$(b, k_b).$$

A DIGITAL SIGNATURE of one BIT

Let us start with a very simple, but much illustrative (though non-practical), example how to sign a single bit.

Design of the signature scheme:

A one-way function f(x) is publicly chosen.

Two integers k_0 and k_1 are chosen and kept **secret** by the signer. Three items

$$f,\ (0,\ s_0),\ (1,\ s_1)$$

are made public, where

$$s_0 = f(k_0),\ s_1 = f(k_1)$$

Signature of a bit b:

$$(\mathrm{b},\ k_b).$$

Verification of such a signature

A DIGITAL SIGNATURE of one BIT

Let us start with a very simple, but much illustrative (though non-practical), example how to sign a single bit.

Design of the signature scheme:

A one-way function f(x) is publicly chosen.

Two integers k_0 and k_1 are chosen and kept **secret** by the signer. Three items

$$f, (0, s_0), (1, s_1)$$

are made public, where

$$s_0 = f(k_0),\ s_1 = f(k_1)$$

Signature of a bit b:

$$(b, k_b).$$

Verification of such a signature

$$s_b = f(k_b)??$$

SECURITY?

FROM RSA CRYPTOSYSTEM to RSA SIGNATURES

The idea of RSA cryptosystem is simple.
Public key: modulus $n = pq$ and encryption exponent e.
Secret key: decryption exponent d and primes p, q

Encryption of a message w: $c = w^e$
Decryption of the cryptotext c: $w = c^d$.

FROM RSA CRYPTOSYSTEM to RSA SIGNATURES

The idea of RSA cryptosystem is simple.
Public key: modulus $n = pq$ and encryption exponent e.
Secret key: decryption exponent d and primes p, q

Encryption of a message w: $c = w^e$
Decryption of the cryptotext c: $w = c^d$.

Does it has a sense to change the order of these two operations: To do first

$$c = w^d$$

and then compute

$$c^e?$$

FROM RSA CRYPTOSYSTEM to RSA SIGNATURES

The idea of RSA cryptosystem is simple.
Public key: modulus $n = pq$ and encryption exponent e.
Secret key: decryption exponent d and primes p, q

Encryption of a message w: $c = w^e$
Decryption of the cryptotext c: $w = c^d$.

Does it has a sense to change the order of these two operations: To do first

$$c = w^d$$

and then compute

$$c^e?$$

Is this a crazy idea?

FROM RSA CRYPTOSYSTEM to RSA SIGNATURES

The idea of RSA cryptosystem is simple.
Public key: modulus $n = pq$ and encryption exponent e.
Secret key: decryption exponent d and primes p, q

Encryption of a message w: $c = w^e$
Decryption of the cryptotext c: $w = c^d$.

Does it has a sense to change the order of these two operations: To do first

$$c = w^d$$

and then compute

$$c^e?$$

Is this a crazy idea? No, we just ned to interpret outcomes of these operations differently.

Indeed,

$$s = w^d$$

should be interpreted as the signature of the message w

FROM RSA CRYPTOSYSTEM to RSA SIGNATURES

The idea of RSA cryptosystem is simple.
Public key: modulus $n = pq$ and encryption exponent e.
Secret key: decryption exponent d and primes p, q

Encryption of a message w: $c = w^e$
Decryption of the cryptotext c: $w = c^d$.

Does it has a sense to change the order of these two operations: To do first

$$c = w^d$$

and then compute

$$c^e?$$

Is this a crazy idea? No, we just ned to interpret outcomes of these operations differently.

Indeed,

$$s = w^d$$

should be interpreted as the signature of the message w

and

$$w = s^e?$$

as a verification of such signature.

RSA SIGNATURES and some ATTACKS on them

Let us have an RSA cryptosystem with encryption and decryption exponents e and d and modulus n.

Signing of a message w:

$$\sigma = w^d \text{ mod n}$$

Verification of the signature $s = \sigma$:

$$w = \sigma^e \text{ mod n?}$$

RSA SIGNATURES and some ATTACKS on them

Let us have an RSA cryptosystem with encryption and decryption exponents e and d and modulus n.

Signing of a message w:

$$\sigma = w^d \bmod n$$

Verification of the signature $s = \sigma$:

$$w = \sigma^e \bmod n?$$

Possible simple attacks

- It might happen that Bob accepts a signature not produced by Alice.

RSA SIGNATURES and some ATTACKS on them

Let us have an RSA cryptosystem with encryption and decryption exponents e and d and modulus n.

Signing of a message w:

$$\sigma = w^d \text{ mod n}$$

Verification of the signature $s = \sigma$:

$$w = \sigma^e \text{ mod n?}$$

Possible simple attacks

- It might happen that Bob accepts a signature not produced by Alice. Indeed, let Eve, using Alice's public key, compute $s = w^e$ for some w and says that w is Alice's signature of s.

  Everybody trying to verify such a signature as Alice's signature gets $w^e = w^e$.

RSA SIGNATURES and some ATTACKS on them

Let us have an RSA cryptosystem with encryption and decryption exponents e and d and modulus n.

Signing of a message w:

$$\sigma = w^d \bmod n$$

Verification of the signature $s = \sigma$:

$$w = \sigma^e \bmod n?$$

Possible simple attacks

- It might happen that Bob accepts a signature not produced by Alice. Indeed, let Eve, using Alice's public key, compute $s = w^e$ for some w and says that w is Alice's signature of s.

  Everybody trying to verify such a signature as Alice's signature gets $w^e = w^e$.

- Some new signatures can be produced without knowing the secret key.

RSA SIGNATURES and some ATTACKS on them

Let us have an RSA cryptosystem with encryption and decryption exponents e and d and modulus n.

Signing of a message w:

$$\sigma = w^d \text{ mod n}$$

Verification of the signature $s = \sigma$:

$$w = \sigma^e \text{ mod n?}$$

Possible simple attacks

- It might happen that Bob accepts a signature not produced by Alice. Indeed, let Eve, using Alice's public key, compute $s = w^e$ for some w and says that w is Alice's signature of s.

  Everybody trying to verify such a signature as Alice's signature gets $w^e = w^e$.

- Some new signatures can be produced without knowing the secret key.

  Indeed, is σ_1 and σ_2 are signatures for w_1 and w_2, then $\sigma_1\sigma_2$ and σ_1^{-1} are signatures for $w_1 w_2$ and w_1^{-1}.

ENCRYPTIONS versus SIGNATURES - SUMMARY

Let each user U use a cryptosystem with encryption and decryption algorithms: e_U, d_U

ENCRYPTIONS versus SIGNATURES - SUMMARY

Let each user U use a cryptosystem with encryption and decryption algorithms: e_U, d_U

Let w be a message

ENCRYPTIONS versus SIGNATURES - SUMMARY

Let each user U use a cryptosystem with encryption and decryption algorithms: e_U, d_U

Let w be a message

PUBLIC-KEY ENCRYPTIONS

Encryption: $e_U(w)$
Decryption: $d_U\ (e_U(w))$

ENCRYPTIONS versus SIGNATURES - SUMMARY

Let each user U use a cryptosystem with encryption and decryption algorithms: e_U, d_U

Let w be a message

PUBLIC-KEY ENCRYPTIONS

Encryption: $e_U(w)$
Decryption: $d_U\ (e_U(w))$

ENCRYPTIONS versus SIGNATURES - SUMMARY

Let each user U use a cryptosystem with encryption and decryption algorithms: e_U, d_U

Let w be a message

PUBLIC-KEY ENCRYPTIONS

Encryption: $e_U(w)$
Decryption: $d_U\ (e_U(w))$

PUBLIC-KEY SIGNATURES

Signing: $d_U(w)$
Verification of the signature: $e_U\ (d_U(w))$

RABIN SIGNATURES

A collision-resistant hash function $h : \{0,1\}^* \rightarrow \{0,1\}^k$ is used for some fixed k.

RABIN SIGNATURES

A collision-resistant hash function $h : \{0,1\}^* \rightarrow \{0,1\}^k$ is used for some fixed k.

Keys generation: The signer S chooses primes p, q of size approximately $k/2$ and computes $n = pq$.

RABIN SIGNATURES

A collision-resistant hash function $h : \{0,1\}^* \rightarrow \{0,1\}^k$ is used for some fixed k.

Keys generation: The signer S chooses primes p, q of size approximately $k/2$ and computes $n = pq$.
n will be the public key
the pair (p, q) will be the secret key.

RABIN SIGNATURES

A collision-resistant hash function $h : \{0,1\}^* \rightarrow \{0,1\}^k$ is used for some fixed k.

Keys generation: The signer S chooses primes p, q of size approximately $k/2$ and computes $n = pq$.
n will be the public key
the pair (p, q) will be the secret key.

Signing:

- To sign a message w, the signer chooses random string U and calculates $h(wU)$;

RABIN SIGNATURES

A collision-resistant hash function $h : \{0,1\}^* \to \{0,1\}^k$ is used for some fixed k.

Keys generation: The signer S chooses primes p, q of size approximately $k/2$ and computes $n = pq$.
n will be the public key
the pair (p, q) will be the secret key.

Signing:

- To sign a message w, the signer chooses random string U and calculates $h(wU)$;
- If $h(wU) \notin QR(n)$, the signer picks a new U and repeats the process;

RABIN SIGNATURES

A collision-resistant hash function $h: \{0,1\}^* \rightarrow \{0,1\}^k$ is used for some fixed k.

Keys generation: The signer S chooses primes p, q of size approximately $k/2$ and computes $n = pq$.
n will be the public key
the pair (p, q) will be the secret key.

Signing:

- To sign a message w, the signer chooses random string U and calculates $h(wU)$;
- If $h(wU) \notin QR(n)$, the signer picks a new U and repeats the process;
- Signer solves the equation $x^2 = h(wU) \mod n$;

RABIN SIGNATURES

A collision-resistant hash function $h : \{0,1\}^* \rightarrow \{0,1\}^k$ is used for some fixed k.

Keys generation: The signer S chooses primes p, q of size approximately $k/2$ and computes $n = pq$.
n will be the public key
the pair (p, q) will be the secret key.

Signing:
- To sign a message w, the signer chooses random string U and calculates $h(wU)$;
- If $h(wU) \notin QR(n)$, the signer picks a new U and repeats the process;
- Signer solves the equation $x^2 = h(wU) \mod n$;
- The pair (U, x) is the signature of w.

RABIN SIGNATURES

A collision-resistant hash function $h : \{0,1\}^* \to \{0,1\}^k$ is used for some fixed k.

Keys generation: The signer S chooses primes p, q of size approximately $k/2$ and computes $n = pq$.
n will be the public key
the pair (p, q) will be the secret key.

Signing:
- To sign a message w, the signer chooses random string U and calculates $h(wU)$;
- If $h(wU) \notin QR(n)$, the signer picks a new U and repeats the process;
- Signer solves the equation $x^2 = h(wU) \mod n$;
- The pair (U, x) is the signature of w.

Verification: Given a message w and a signature (U, x) the versifier V computes x^2 and $h(wU)$ and verifies that they are equal.

IMPORTANT FACTS

Fact 1

If, for integers a, b and a prime p,

$$a \equiv b \pmod{(p-1)}$$

then for any integer x

$$x^a \equiv x^b (\text{mod } p)$$

IMPORTANT FACTS

Fact 1

If, for integers a, b and a prime p,

$$a \equiv b \pmod{(p-1)}$$

then for any integer x

$$x^a \equiv x^b \pmod{p}$$

Fact 2

If a, b, n, x are integers and $gcd(x, n) = 1$, then

$$a \equiv b \pmod{\phi(n)} \text{ implies } x^a \equiv x^b \pmod{n}$$

Let

$$a \equiv b \mod (p-1)$$

PROOF

Let

$$a \equiv b \mod (p-1)$$

then

$$x^a = x^{k(p-1)+b}$$

for some k, any x and therefore

PROOF

Let

$$a \equiv b \mod (p-1)$$

then

$$x^a = x^{k(p-1)+b}$$

for some k, any x and therefore

$$x^a = x^b (x^{p-1})^k \equiv x^b \mod p$$

by Fermat's little theorem.

ElGamal SIGNATURES

ElGamal SIGNATURES

Design of the ElGamal digital signature system: **choose**: prime p, integers $1 \leq q \leq x \leq p$, where q is a primitive element of Z_p^*;

ElGamal SIGNATURES

Design of the ElGamal digital signature system: **choose**: prime p, integers $1 \leq q \leq x \leq p$, where q is a primitive element of Z_p^*;

Compute: $y = q^x \mod p$

ElGamal SIGNATURES

Design of the ElGamal digital signature system: **choose**: prime p, integers $1 \leq q \leq x \leq p$, where q is a primitive element of Z_p^*;

$$\textbf{Compute}: \; y = q^x \mod p$$

$$\textbf{key K} = (\mathrm{p, q, x, y})$$

ElGamal SIGNATURES

Design of the ElGamal digital signature system: **choose**: prime p, integers $1 \leq q \leq x \leq p$, where q is a primitive element of Z_p^*;

Compute: $y = q^x \mod p$

key K $=$ (p, q, x, y)

public key (p, q, y) - **secret key**: x

ElGamal SIGNATURES

Design of the ElGamal digital signature system: **choose**: prime p, integers $1 \leq q \leq x \leq p$, where q is a primitive element of Z_p^*;

$$\textbf{Compute}: \; y = q^x \mod p$$

$$\textbf{key K} = (\text{p, q, x, y})$$

$$\textbf{public key} \; (\text{p, q, y}) \text{ - } \textbf{secret key}: \text{x}$$

Signature of a message w: Let $r \in Z_{p-1}^*$ be randomly chosen and kept secret.

$$\text{sig(w, r)} = \text{(a, b)},$$

$$\text{where } a = q^r \mod p$$

$$\text{and b} = (w - xa)r^{-1} \pmod{(p-1)}.$$

ElGamal SIGNATURES

Design of the ElGamal digital signature system: **choose**: prime p, integers $1 \leq q \leq x \leq p$, where q is a primitive element of Z_p^*;

$$\textbf{Compute}: \ y = q^x \mod p$$

$$\textbf{key K} = (p, q, x, y)$$

public key (p, q, y) - **secret key**: x

Signature of a message w: Let $r \in Z_{p-1}^*$ be randomly chosen and kept secret.

$$\text{sig}(w, r) = (a, b),$$

$$\text{where } a = q^r \mod p$$

$$\text{and } b = (w - xa)r^{-1} \pmod{(p-1)}.$$

Verification: accept a signature (a,b) of w as valid if

$$y^a a^b = q^w \pmod p$$

ElGamal SIGNATURES

Design of the ElGamal digital signature system: **choose**: prime p, integers $1 \leq q \leq x \leq p$, where q is a primitive element of Z_p^*;

$$\textbf{Compute}: \; y = q^x \mod p$$

$$\textbf{key K} = (p, q, x, y)$$

public key (p, q, y) - **secret key**: x

Signature of a message w: Let $r \in Z_{p-1}^*$ be randomly chosen and kept secret.

$$\text{sig}(w, r) = (a, b),$$

$$\text{where } a = q^r \mod p$$

$$\text{and } b = (w - xa)r^{-1} \pmod{(p-1)}.$$

Verification: accept a signature (a,b) of w as valid if

$$y^a a^b = q^w \pmod p$$

(Indeed, for some integer k: $y^a a^b \equiv q^{ax} q^{rb} \equiv q^{ax+w-ax+k(p-1)} \equiv q^w \pmod p$)

SECURITY of ElGamal SIGNATURES

Let us analyze several ways an eavesdropper Eve can try to forge ElGamal signature (with x - secret; p, q and y = q^x mod p - public):

$$\text{sig}(w, r) = (a, b);$$

where r is random and a = q^r mod p; b = $(w - xa)r^{-1} \pmod{p-1}$.

SECURITY of ElGamal SIGNATURES

Let us analyze several ways an eavesdropper Eve can try to forge ElGamal signature (with x - secret; p, q and y = q^x mod p - public):

$$\text{sig}(w, r) = (a, b);$$

where r is random and a = q^r mod p; b = $(w - xa)r^{-1} \pmod{p-1}$.

1. First suppose Eve tries to forge signature for a new message w, without knowing x.
   - If Eve first chooses a value a and tries to find the corresponding b, it has to compute the discrete logarithm

     $$lg_a q^w y^{-a},$$

     (because $a^b \equiv q^{r(w-xa)r^{-1}} \equiv q^{w-xa} \equiv q^w y^{-a}$) what is infeasible.
   - If Eve first chooses **b** and then tries to find **a**, she has to solve the equation

     $$y^a a^b \equiv q^{xa} q^{rb} \equiv q^w \pmod{p}.$$

     It is not known whether this equation can be solved for any given b efficiently.

SECURITY of ElGamal SIGNATURES

Let us analyze several ways an eavesdropper Eve can try to forge ElGamal signature (with x - secret; p, q and y = q^x mod p - public):

$$\text{sig}(w, r) = (a, b);$$

where r is random and a = q^r mod p; b = $(w - xa)r^{-1} \pmod{p-1}$.

1. First suppose Eve tries to forge signature for a new message w, without knowing x.
   - If Eve first chooses a value a and tries to find the corresponding b, it has to compute the discrete logarithm

     $$lg_a q^w y^{-a},$$

     (because $a^b \equiv q^{r(w-xa)r^{-1}} \equiv q^{w-xa} \equiv q^w y^{-a}$) what is infeasible.
   - If Eve first chooses **b** and then tries to find **a**, she has to solve the equation

     $$y^a a^b \equiv q^{xa} q^{rb} \equiv q^w \pmod{p}.$$

     It is not known whether this equation can be solved for any given b efficiently.
2. If Eve chooses **a** and **b** and tries to determine **w** such that (a,b) is signature of w, then she has to compute discrete logarithm

   $$lg_q y^a a^b.$$

   Hence, Eve can not sign a "random" message this way.

From ElGamal to DSA (DIGITAL SIGNATURE STANDARD)

From ElGamal to DSA (DIGITAL SIGNATURE STANDARD)

DSA is a **digital signature standard**, described on the next two slides, that is a modification of ElGamal digital signature scheme.

From ElGamal to DSA (DIGITAL SIGNATURE STANDARD)

DSA is a **digital signature standard**, described on the next two slides, that is a modification of ElGamal digital signature scheme. It was proposed in August 1991 and adopted in December 1994.

From ElGamal to DSA (DIGITAL SIGNATURE STANDARD)

DSA is a **digital signature standard**, described on the next two slides, that is a modification of ElGamal digital signature scheme. It was proposed in August 1991 and adopted in December 1994.

Any proposal for digital signature standard has to go through a very careful scrutiny. Why?

From ElGamal to DSA (DIGITAL SIGNATURE STANDARD)

DSA is a **digital signature standard**, described on the next two slides, that is a modification of ElGamal digital signature scheme. It was proposed in August 1991 and adopted in December 1994.

Any proposal for digital signature standard has to go through a very careful scrutiny. Why?

Encryption of a message is usually done only once and therefore it usually suffices to use a cryptosystem that is secure **at the time of the encryption.**

On the other hand, a signed message could be a contract or a will and it can happen that it will be needed to verify its signature **many years after the message is signed.**

From ElGamal to DSA (DIGITAL SIGNATURE STANDARD)

DSA is a **digital signature standard**, described on the next two slides, that is a modification of ElGamal digital signature scheme. It was proposed in August 1991 and adopted in December 1994.

Any proposal for digital signature standard has to go through a very careful scrutiny. Why?

Encryption of a message is usually done only once and therefore it usually suffices to use a cryptosystem that is secure **at the time of the encryption.**

On the other hand, a signed message could be a contract or a will and it can happen that it will be needed to verify its signature **many years after the message is signed.**

Since ElGamal signature is no more secure than discrete logarithm, it is necessary to use large p, with at least 512 bits.

From ElGamal to DSA (DIGITAL SIGNATURE STANDARD)

DSA is a **digital signature standard**, described on the next two slides, that is a modification of ElGamal digital signature scheme. It was proposed in August 1991 and adopted in December 1994.

Any proposal for digital signature standard has to go through a very careful scrutiny. Why?

Encryption of a message is usually done only once and therefore it usually suffices to use a cryptosystem that is secure **at the time of the encryption.**

On the other hand, a signed message could be a contract or a will and it can happen that it will be needed to verify its signature **many years after the message is signed.**

Since ElGamal signature is no more secure than discrete logarithm, it is necessary to use large p, with at least 512 bits.

However, with ElGamal this would lead to signatures with at least 1024 bits what is too much for such applications as smart cards.

DIGITAL SIGNATURE STANDARD I

In December 1994, on the proposal of the National Institute of Standards and Technology, the following Digital Signature Algorithm (DSA) was accepted as a standard.

DIGITAL SIGNATURE STANDARD I

In December 1994, on the proposal of the National Institute of Standards and Technology, the following Digital Signature Algorithm (DSA) was accepted as a standard.

Design of DSA

1. **The following global public key components** are chosen:
   - p - a random l-bit prime, $512 \leq l \leq 1024$, l = 64k.
   - q - a random 160-bit prime dividing p -1.
   - r = $h^{(p-1)/q} \mod p$, where h is a random primitive element of Z_p, such that $r > 1$, $r \neq 1$ (observe that r is a q-th root of 1 mod p).

DIGITAL SIGNATURE STANDARD I

In December 1994, on the proposal of the National Institute of Standards and Technology, the following Digital Signature Algorithm (DSA) was accepted as a standard.

Design of DSA

1. **The following global public key components** are chosen:
   - p - a random l-bit prime, $512 \leq l \leq 1024$, l = 64k.
   - q - a random 160-bit prime dividing p -1.
   - r = $h^{(p-1)/q} \mod p$, where h is a random primitive element of Z_p, such that $r > 1$, $r \neq 1$ (observe that r is a q-th root of 1 mod p).
2. **The following user's private key component** is chosen:
   - x - a random integer (once), $0 < x < q$,
3. The following value is also made public
   - y = r^x mod p.

DIGITAL SIGNATURE STANDARD I

In December 1994, on the proposal of the National Institute of Standards and Technology, the following Digital Signature Algorithm (DSA) was accepted as a standard.

Design of DSA

1. **The following global public key components** are chosen:
   - p - a random l-bit prime, $512 \leq l \leq 1024$, l = 64k.
   - q - a random 160-bit prime dividing p -1.
   - r = $h^{(p-1)/q} \mod p$, where h is a random primitive element of Z_p, such that $r > 1$, $r \neq 1$ (observe that r is a q-th root of 1 mod p).
2. **The following user's private key component** is chosen:
   - x - a random integer (once), $0 < x < q$,
3. The following value is also made public
   - y = r^x mod p.
4. Key is K = (p, q, r, x, y)

Signing and Verification

Signing of a 160-bit plaintext w

- choose random $0 < k < q$
- compute a $= (r^k \mod p) \bmod q$
- compute b $= k^{-1}(w + xa) \bmod q$ where $kk^{-1} \equiv 1 \pmod q$
- signature: sig(w, k) = (a, b)

Signing and Verification

Signing of a 160-bit plaintext w

- choose random $0 < k < q$
- compute a $= (r^k \mod p) \text{ mod q}$
- compute b $= k^{-1}(\text{w} + \text{xa}) \text{ mod q where } kk^{-1} \equiv 1 \pmod q$
- signature: sig(w, k) = (a, b)

Verification of signature (a, b)

- compute $z = b^{-1} \mod q$
- compute $u_1 = wz \mod q$, $u_2 = az \mod q$

  verification:

$$ver_K(w, a, b) = true \Leftrightarrow (r^{u_1} y^{u_2} \mod p) \mod q = a$$

From ElGamal to DSA - II

In DSA a 160 bit message is signed using 320-bit signature, but computation is done modulo with 512-1024 bits.

From ElGamal to DSA - II

In DSA a 160 bit message is signed using 320-bit signature, but computation is done modulo with 512-1024 bits.

Observe that y and a are also q-roots of 1. Hence any exponents of r,y and a can be reduced modulo q without affecting the verification condition.

This allowed to change ElGamal verification condition: $y^a a^b = q^w$.

Fiat-Shamir SIGNATURE SCHEME

Fiat-Shamir SIGNATURE SCHEME

Choose primes p, q, compute n = pq and choose: as a **public key** integers $v_1, \ldots, v_k$ and compute, as a **secret key**, $s_1, \ldots, s_k, s_i = \sqrt{v_i^{-1}} \mod n$.

Fiat-Shamir SIGNATURE SCHEME

Choose primes p, q, compute n = pq and choose: as a **public key** integers $v_1, \ldots, v_k$ and compute, as a **secret key**, $s_1, \ldots, s_k, s_i = \sqrt{v_i^{-1}} \mod n$.

Protocol for Alice **to sign a message** w:

1. Alice first chooses (as a security parameter) an integer t, then t random integers $1 \leq r_1, \ldots, r_t < n$, and computes $x_i = r_i^2 \mod n$, for $1 \leq i \leq t$.

Fiat-Shamir SIGNATURE SCHEME

Choose primes p, q, compute n = pq and choose: as a **public key** integers $v_1, \ldots, v_k$ and compute, as a **secret key**, $s_1, \ldots, s_k, s_i = \sqrt{v_i^{-1}} \mod n$.

Protocol for Alice **to sign a message** w:

1. Alice first chooses (as a security parameter) an integer t, then t random integers $1 \leq r_1, \ldots, r_t < n$, and computes $x_i = r_i^2 \mod n$, for $1 \leq i \leq t$.
2. Alice uses a publicly known hash function h to compute $H = h(wx_1x_2 \ldots x_t)$ and then uses the first kt bits of H, denoted as b_{ij}, $1 \leq i \leq t, 1 \leq j \leq k$ as follows.

Fiat-Shamir SIGNATURE SCHEME

Choose primes p, q, compute n = pq and choose: as a **public key** integers $v_1, \ldots, v_k$ and compute, as a **secret key**, $s_1, \ldots, s_k, s_i = \sqrt{v_i^{-1}} \mod n$.

Protocol for Alice **to sign a message** w:

1. Alice first chooses (as a security parameter) an integer t, then t random integers $1 \leq r_1, \ldots, r_t < n$, and computes $x_i = r_i^2 \mod n$, for $1 \leq i \leq t$.
2. Alice uses a publicly known hash function h to compute $H = h(wx_1x_2 \ldots x_t)$ and then uses the first kt bits of H, denoted as b_{ij}, $1 \leq i \leq t, 1 \leq j \leq k$ as follows.
3. Alice computes $y_1, \ldots, y_t$

$$y_i = r_i \prod_{j=1}^{k} s_j^{b_{ij}} \mod n$$

Fiat-Shamir SIGNATURE SCHEME

Choose primes p, q, compute n = pq and choose: as a **public key** integers $v_1, \dots, v_k$ and compute, as a **secret key**, $s_1, \dots, s_k, s_i = \sqrt{v_i^{-1}} \mod n$.

Protocol for Alice **to sign a message** w:

1. Alice first chooses (as a security parameter) an integer t, then t random integers $1 \le r_1, \dots, r_t < n$, and computes $x_i = r_i^2 \mod n$, for $1 \le i \le t$.
2. Alice uses a publicly known hash function h to compute $H = h(wx_1x_2 \dots x_t)$ and then uses the first kt bits of H, denoted as b_{ij}, $1 \le i \le t, 1 \le j \le k$ as follows.
3. Alice computes $y_1, \dots, y_t$
$$y_i = r_i \prod_{j=1}^{k} s_j^{b_{ij}} \mod n$$
4. Alice sends to Bob w, all b_{ij}, all y_i and also h {Bob already knows Alice's public key $v_1, \dots, v_k$}

Fiat-Shamir SIGNATURE SCHEME

Choose primes p, q, compute n = pq and choose: as a **public key** integers $v_1, \dots, v_k$ and compute, as a **secret key**, $s_1, \dots, s_k, s_i = \sqrt{v_i^{-1}} \mod n$.

Protocol for Alice **to sign a message** w:

1. Alice first chooses (as a security parameter) an integer t, then t random integers $1 \le r_1, \dots, r_t < n$, and computes $x_i = r_i^2 \mod n$, for $1 \le i \le t$.
2. Alice uses a publicly known hash function h to compute $H = h(wx_1x_2 \dots x_t)$ and then uses the first kt bits of H, denoted as b_{ij}, $1 \le i \le t, 1 \le j \le k$ as follows.
3. Alice computes $y_1, \dots, y_t$
$$y_i = r_i \prod_{j=1}^{k} s_j^{b_{ij}} \mod n$$
4. Alice sends to Bob w, all b_{ij}, all y_i and also h {Bob already knows Alice's public key $v_1, \dots, v_k$}
5. Bob finally computes $z_1, \dots, z_k$, where
$$z_i = y_i^2 \prod_{j=1}^{k} v_j^{b_{ij}} \mod n$$

Fiat-Shamir SIGNATURE SCHEME

Choose primes p, q, compute n = pq and choose: as a **public key** integers $v_1, \ldots, v_k$ and compute, as a **secret key**, $s_1, \ldots, s_k, s_i = \sqrt{v_i^{-1}} \mod n$.

Protocol for Alice **to sign a message** w:

1. Alice first chooses (as a security parameter) an integer t, then t random integers $1 \leq r_1, \ldots, r_t < n$, and computes $x_i = r_i^2 \mod n$, for $1 \leq i \leq t$.
2. Alice uses a publicly known hash function h to compute $H = h(wx_1x_2 \ldots x_t)$ and then uses the first kt bits of H, denoted as b_{ij}, $1 \leq i \leq t, 1 \leq j \leq k$ as follows.
3. Alice computes $y_1, \ldots, y_t$
$$y_i = r_i \prod_{j=1}^{k} s_j^{b_{ij}} \mod n$$
4. Alice sends to Bob w, all b_{ij}, all y_i and also h {Bob already knows Alice's public key $v_1, \ldots, v_k$}
5. Bob finally computes $z_1, \ldots, z_k$, where
$$z_i = y_i^2 \prod_{j=1}^{k} v_j^{b_{ij}} \mod n = r_i^2 \prod_{j=1}^{k} (v_j^{-1})^{b_{ij}} \prod_{j=1}^{k} v_j^{b_{ij}} =$$

Fiat-Shamir SIGNATURE SCHEME

Choose primes p, q, compute n = pq and choose: as a **public key** integers $v_1, \dots, v_k$ and compute, as a **secret key**, $s_1, \dots, s_k, s_i = \sqrt{v_i^{-1}} \mod n$.

Protocol for Alice **to sign a message** w:

1. Alice first chooses (as a security parameter) an integer t, then t random integers $1 \leq r_1, \dots, r_t < n$, and computes $x_i = r_i^2 \mod n$, for $1 \leq i \leq t$.
2. Alice uses a publicly known hash function h to compute $H = h(wx_1x_2 \dots x_t)$ and then uses the first kt bits of H, denoted as b_{ij}, $1 \leq i \leq t, 1 \leq j \leq k$ as follows.
3. Alice computes $y_1, \dots, y_t$
$$y_i = r_i \prod_{j=1}^{k} s_j^{b_{ij}} \mod n$$
4. Alice sends to Bob w, all b_{ij}, all y_i and also h {Bob already knows Alice's public key $v_1, \dots, v_k$}
5. Bob finally computes $z_1, \dots, z_k$, where
$$z_i = y_i^2 \prod_{j=1}^{k} v_j^{b_{ij}} \mod n = r_i^2 \prod_{j=1}^{k} (v_j^{-1})^{b_{ij}} \prod_{j=1}^{k} v_j^{b_{ij}} = r_i^2 = x_i$$
and verifies that the first $k \times t$ bits of $h(wx_1x_2 \dots x_t)$ are the b_{ij} values that Alice has sent to him.

Fiat-Shamir SIGNATURE SCHEME

Choose primes p, q, compute n = pq and choose: as a **public key** integers $v_1, \ldots, v_k$ and compute, as a **secret key**, $s_1, \ldots, s_k, s_i = \sqrt{v_i^{-1}} \mod n$.

Protocol for Alice **to sign a message** w:

1. Alice first chooses (as a security parameter) an integer t, then t random integers $1 \leq r_1, \ldots, r_t < n$, and computes $x_i = r_i^2 \mod n$, for $1 \leq i \leq t$.
2. Alice uses a publicly known hash function h to compute $H = h(wx_1x_2 \ldots x_t)$ and then uses the first kt bits of H, denoted as b_{ij}, $1 \leq i \leq t, 1 \leq j \leq k$ as follows.
3. Alice computes $y_1, \ldots, y_t$
$$y_i = r_i \prod_{j=1}^{k} s_j^{b_{ij}} \mod n$$
4. Alice sends to Bob w, all b_{ij}, all y_i and also h {Bob already knows Alice's public key $v_1, \ldots, v_k$}
5. Bob finally computes $z_1, \ldots, z_k$, where
$$z_i = y_i^2 \prod_{j=1}^{k} v_j^{b_{ij}} \mod n = r_i^2 \prod_{j=1}^{k} (v_j^{-1})^{b_{ij}} \prod_{j=1}^{k} v_j^{b_{ij}} = r_i^2 = x_i$$
and verifies that the first $k \times t$ bits of $h(wx_1x_2 \ldots x_t)$ are the b_{ij} values that Alice has sent to him.

   Security of this signature scheme is 2^{-kt}.

   Advantage over the RSA-based signature scheme: only about 5% of modular multiplications are needed.

SAD STORY

SAD STORY

Alice and Bob got to jail - and, unfortunately, to different jails.

Alice and Bob got to jail - and, unfortunately, to different jails.

Walter, the warden, allows them to communicate by network, but he will not allow their messages to be encrypted.

SAD STORY

Alice and Bob got to jail - and, unfortunately, to different jails.

Walter, the warden, allows them to communicate by network, but he will not allow their messages to be encrypted.

Problem: Can Alice and Bob set up a subliminal channel, a covert communication channel between them, in full view of Walter, even though the messages themselves that they exchange contain no secret information?

Ong-Schnorr-Shamir SUBLUMINAL CHANNEL SCHEME

Story Alice and Bob are in different jails. Walter, the warden, allows them to communicate by network, but he will not allow messages to be encrypted. Can they set up a subliminal channel, a covert communication channel between them, in full view of Walter, even though the messages themselves contain no secret information?

Ong-Schnorr-Shamir SUBLUMINAL CHANNEL SCHEME

Story Alice and Bob are in different jails. Walter, the warden, allows them to communicate by network, but he will not allow messages to be encrypted. Can they set up a subliminal channel, a covert communication channel between them, in full view of Walter, even though the messages themselves contain no secret information?

Yes. Alice and Bob create first the following communication scheme:

Ong-Schnorr-Shamir SUBLUMINAL CHANNEL SCHEME

Story Alice and Bob are in different jails. Walter, the warden, allows them to communicate by network, but he will not allow messages to be encrypted. Can they set up a subliminal channel, a covert communication channel between them, in full view of Walter, even though the messages themselves contain no secret information?

Yes. Alice and Bob create first the following communication scheme:

They choose a large n and an integer k such that **$\gcd(n, k) = 1$**.

Ong-Schnorr-Shamir SUBLUMINAL CHANNEL SCHEME

Story Alice and Bob are in different jails. Walter, the warden, allows them to communicate by network, but he will not allow messages to be encrypted. Can they set up a subliminal channel, a covert communication channel between them, in full view of Walter, even though the messages themselves contain no secret information?

Yes. Alice and Bob create first the following communication scheme:

They choose a large n and an integer k such that **gcd(n, k) = 1**.

They calculate $h = k^{-2} \mod n = (k^{-1})^2 \mod n$.

Ong-Schnorr-Shamir SUBLUMINAL CHANNEL SCHEME

Story Alice and Bob are in different jails. Walter, the warden, allows them to communicate by network, but he will not allow messages to be encrypted. Can they set up a subliminal channel, a covert communication channel between them, in full view of Walter, even though the messages themselves contain no secret information?

Yes. Alice and Bob create first the following communication scheme:

They choose a large n and an integer k such that **gcd(n, k) = 1**.

They calculate $h = k^{-2} \mod n = (k^{-1})^2 \mod n$.

They make h, n to be **public key**

Ong-Schnorr-Shamir SUBLUMINAL CHANNEL SCHEME

Story Alice and Bob are in different jails. Walter, the warden, allows them to communicate by network, but he will not allow messages to be encrypted. Can they set up a subliminal channel, a covert communication channel between them, in full view of Walter, even though the messages themselves contain no secret information?

Yes. Alice and Bob create first the following communication scheme:

They choose a large n and an integer k such that **gcd(n, k) = 1**.

They calculate $h = k^{-2} \mod n = (k^{-1})^2 \mod n$.

They make h, n to be **public key**

They keep secret k as trapdoor information.

Ong-Schnorr-Shamir SUBLUMINAL CHANNEL SCHEME

Story Alice and Bob are in different jails. Walter, the warden, allows them to communicate by network, but he will not allow messages to be encrypted. Can they set up a subliminal channel, a covert communication channel between them, in full view of Walter, even though the messages themselves contain no secret information?

Yes. Alice and Bob create first the following communication scheme:

They choose a large n and an integer k such that **gcd(n, k) = 1**.

They calculate $h = k^{-2} \mod n = (k^{-1})^2 \mod n$.

They make h, n to be **public key**

They keep secret k as trapdoor information.

Let **w** be secret message Alice wants to send (it has to be such that gcd(w, n) =1)

Ong-Schnorr-Shamir SUBLUMINAL CHANNEL SCHEME

Story Alice and Bob are in different jails. Walter, the warden, allows them to communicate by network, but he will not allow messages to be encrypted. Can they set up a subliminal channel, a covert communication channel between them, in full view of Walter, even though the messages themselves contain no secret information?

Yes. Alice and Bob create first the following communication scheme:

They choose a large n and an integer k such that **gcd(n, k) = 1**.

They calculate $h = k^{-2} \mod n = (k^{-1})^2 \mod n$.

They make h, n to be **public key**

They keep secret k as trapdoor information.

Let **w** be secret message Alice wants to send (it has to be such that gcd(w, n) =1)
Denote a harmless message she uses by w' (it has to be such that gcd(w ',n) = 1)

Ong-Schnorr-Shamir SUBLUMINAL CHANNEL SCHEME

Story Alice and Bob are in different jails. Walter, the warden, allows them to communicate by network, but he will not allow messages to be encrypted. Can they set up a subliminal channel, a covert communication channel between them, in full view of Walter, even though the messages themselves contain no secret information?

Yes. Alice and Bob create first the following communication scheme:

They choose a large n and an integer k such that **gcd(n, k) = 1**.

They calculate $h = k^{-2} \mod n = (k^{-1})^2 \mod n$.

They make h, n to be **public key**

They keep secret k as trapdoor information.

Let **w** be secret message Alice wants to send (it has to be such that gcd(w, n) =1)
Denote a harmless message she uses by w' (it has to be such that gcd(w ',n) = 1)
Signing by Alice:

Ong-Schnorr-Shamir SUBLUMINAL CHANNEL SCHEME

Story Alice and Bob are in different jails. Walter, the warden, allows them to communicate by network, but he will not allow messages to be encrypted. Can they set up a subliminal channel, a covert communication channel between them, in full view of Walter, even though the messages themselves contain no secret information?

Yes. Alice and Bob create first the following communication scheme:

They choose a large n and an integer k such that **gcd(n, k) = 1**.

They calculate $h = k^{-2} \mod n = (k^{-1})^2 \mod n$.

They make h, n to be **public key**

They keep secret k as trapdoor information.

Let **w** be secret message Alice wants to send (it has to be such that gcd(w, n) =1)
Denote a harmless message she uses by w' (it has to be such that gcd(w ',n) = 1)

Signing by Alice:

$$S_1 = \frac{1}{2} \cdot \left(\frac{w'}{w} + w\right) \mod n$$
$$S_2 = \frac{k}{2} \cdot \left(\frac{w'}{w} - w\right) \mod n$$

Ong-Schnorr-Shamir SUBLUMINAL CHANNEL SCHEME

Story Alice and Bob are in different jails. Walter, the warden, allows them to communicate by network, but he will not allow messages to be encrypted. Can they set up a subliminal channel, a covert communication channel between them, in full view of Walter, even though the messages themselves contain no secret information?

Yes. Alice and Bob create first the following communication scheme:

They choose a large n and an integer k such that **gcd(n, k) = 1**.

They calculate $h = k^{-2} \mod n = (k^{-1})^2 \mod n$.

They make h, n to be **public key**

They keep secret k as trapdoor information.

Let **w** be secret message Alice wants to send (it has to be such that gcd(w, n) =1)
Denote a harmless message she uses by w' (it has to be such that gcd(w ',n) = 1)

Signing by Alice:

$$S_1 = \frac{1}{2} \cdot \left(\frac{w'}{w} + w\right) \mod n$$
$$S_2 = \frac{k}{2} \cdot \left(\frac{w'}{w} - w\right) \mod n$$

Signature: (S_1, S_2). Alice then sends to Bob (w', S_1, S_2)

Ong-Schnorr-Shamir SUBLUMINAL CHANNEL SCHEME

Story Alice and Bob are in different jails. Walter, the warden, allows them to communicate by network, but he will not allow messages to be encrypted. Can they set up a subliminal channel, a covert communication channel between them, in full view of Walter, even though the messages themselves contain no secret information?

Yes. Alice and Bob create first the following communication scheme:

They choose a large n and an integer k such that **gcd(n, k) = 1**.

They calculate $h = k^{-2} \mod n = (k^{-1})^2 \mod n$.

They make h, n to be **public key**

They keep secret k as trapdoor information.

Let **w** be secret message Alice wants to send (it has to be such that gcd(w, n) =1)
Denote a harmless message she uses by w' (it has to be such that gcd(w ',n) = 1)
Signing by Alice:

$$S_1 = \frac{1}{2} \cdot \left(\frac{w'}{w} + w\right) \mod n$$
$$S_2 = \frac{k}{2} \cdot \left(\frac{w'}{w} - w\right) \mod n$$

Signature: (S_1, S_2). Alice then sends to Bob (w', S_1, S_2)
Signature verification method for Walter: w' $= S_1^2 - hS_2^2 (\mod n)$

Ong-Schnorr-Shamir SUBLUMINAL CHANNEL SCHEME

Story Alice and Bob are in different jails. Walter, the warden, allows them to communicate by network, but he will not allow messages to be encrypted. Can they set up a subliminal channel, a covert communication channel between them, in full view of Walter, even though the messages themselves contain no secret information?

Yes. Alice and Bob create first the following communication scheme:

They choose a large n and an integer k such that **gcd(n, k) = 1**.

They calculate $h = k^{-2} \mod n = (k^{-1})^2 \mod n$.

They make h, n to be **public key**

They keep secret k as trapdoor information.

Let **w** be secret message Alice wants to send (it has to be such that gcd(w, n) =1)
Denote a harmless message she uses by w' (it has to be such that gcd(w ',n) = 1)

Signing by Alice:

$$S_1 = \frac{1}{2} \cdot \left(\frac{w'}{w} + w\right) \mod n$$
$$S_2 = \frac{k}{2} \cdot \left(\frac{w'}{w} - w\right) \mod n$$

Signature: (S_1, S_2). Alice then sends to Bob (w', S_1, S_2)
Signature verification method for Walter: w' $= S_1^2 - hS_2^2 (\mod n)$

Decryption by Bob: $w = \dfrac{w'}{(S_1 + k^{-1}S_2)} \mod n$

LAMPORT ONE-TIME SIGNATURES

LAMPORT ONE-TIME SIGNATURES

Lamport signature scheme shows how to construct **a signature scheme for one use only** - from any cryptographically secure one-way function.

LAMPORT ONE-TIME SIGNATURES

Lamport signature scheme shows how to construct **a signature scheme for one use only** - from any cryptographically secure one-way function.

Let **k** be a positive integer and let $M = \{0, 1\}^k$ be the set of messages.

LAMPORT ONE-TIME SIGNATURES

Lamport signature scheme shows how to construct **a signature scheme for one use only** - from any cryptographically secure one-way function.

Let **k** be a positive integer and let $M = \{0,1\}^k$ be the set of messages.
Let **f**: $Y \rightarrow Z$ be a one-way function where Y is a set of "signatures".

LAMPORT ONE-TIME SIGNATURES

Lamport signature scheme shows how to construct **a signature scheme for one use only** - from any cryptographically secure one-way function.

Let **k** be a positive integer and let $M = \{0,1\}^k$ be the set of messages.
Let **f**: $Y \rightarrow Z$ be a one-way function where Y is a set of "signatures".

For $1 \leq i \leq k$, j = 0,1 let $y_{ij} \in Y$ be chosen randomly and $z_{ij} = f(y_{ij})$.

LAMPORT ONE-TIME SIGNATURES

Lamport signature scheme shows how to construct **a signature scheme for one use only** - from any cryptographically secure one-way function.

Let **k** be a positive integer and let $M = \{0, 1\}^k$ be the set of messages.
Let **f**: $Y \rightarrow Z$ be a one-way function where Y is a set of "signatures".

For $1 \leq i \leq k$, j = 0,1 let $y_{ij} \in Y$ be chosen randomly and $z_{ij} = f(y_{ij})$.

The **key K** consists of 2k y's and z's.

LAMPORT ONE-TIME SIGNATURES

Lamport signature scheme shows how to construct **a signature scheme for one use only** - from any cryptographically secure one-way function.

Let **k** be a positive integer and let $M = \{0, 1\}^k$ be the set of messages.
Let **f**: $Y \to Z$ be a one-way function where Y is a set of "signatures".

For $1 \leq i \leq k$, j = 0,1 let $y_{ij} \in Y$ be chosen randomly and $z_{ij} = f(y_{ij})$.

The **key K** consists of 2k y's and z's. y's form the secret key, z's form the public key.

LAMPORT ONE-TIME SIGNATURES

Lamport signature scheme shows how to construct **a signature scheme for one use only** - from any cryptographically secure one-way function.

Let **k** be a positive integer and let $M = \{0,1\}^k$ be the set of messages.
Let **f**: $Y \rightarrow Z$ be a one-way function where Y is a set of "signatures".

For $1 \leq i \leq k$, j = 0,1 let $y_{ij} \in Y$ be chosen randomly and $z_{ij} = f(y_{ij})$.

The **key K** consists of 2k y's and z's. y's form the secret key, z's form the public key.

Signing of a message $x = x_1 \ldots x_k \in \{0,1\}^k$

$$sign(x_1 \ldots x_k) = (y_{1,x_1}, \ldots, y_{k,x_k})$$

LAMPORT ONE-TIME SIGNATURES

Lamport signature scheme shows how to construct **a signature scheme for one use only** - from any cryptographically secure one-way function.

Let **k** be a positive integer and let $M = \{0,1\}^k$ be the set of messages.
Let **f**: $Y \rightarrow Z$ be a one-way function where Y is a set of "signatures".

For $1 \leq i \leq k$, j = 0,1 let $y_{ij} \in Y$ be chosen randomly and $z_{ij} = f(y_{ij})$.

The **key K** consists of 2k y's and z's. y's form the secret key, z's form the public key.

Signing of a message $x = x_1 \dots x_k \in \{0,1\}^k$

$$sign(x_1 \dots x_k) = (y_{1,x_1}, \dots, y_{k,x_k}) = (a_1, \dots, a_k) \text{ - notation}$$

LAMPORT ONE-TIME SIGNATURES

Lamport signature scheme shows how to construct **a signature scheme for one use only** - from any cryptographically secure one-way function.

Let **k** be a positive integer and let $M = \{0, 1\}^k$ be the set of messages.
Let **f**: $Y \rightarrow Z$ be a one-way function where Y is a set of "signatures".

For $1 \leq i \leq k$, j = 0,1 let $y_{ij} \in Y$ be chosen randomly and $z_{ij} = f(y_{ij})$.

The **key K** consists of 2k y's and z's. y's form the secret key, z's form the public key.

Signing of a message $x = x_1 \dots x_k \in \{0, 1\}^k$

$$sign(x_1 \dots x_k) = (y_{1,x_1}, \dots, y_{k,x_k}) = (a_1, \dots, a_k) \text{ - notation}$$

and

$$verif(x_1 \dots x_k, a_1, \dots, a_k) = true \Leftrightarrow f(a_i) = z_{i,xi}, 1 \leq i \leq k$$

Eve cannot forge a signature because she is unable to invert one-way functions.

LAMPORT ONE-TIME SIGNATURES

Lamport signature scheme shows how to construct **a signature scheme for one use only** - from any cryptographically secure one-way function.

Let **k** be a positive integer and let $M = \{0,1\}^k$ be the set of messages.
Let **f**: $Y \rightarrow Z$ be a one-way function where Y is a set of "signatures".

For $1 \leq i \leq k$, j = 0,1 let $y_{ij} \in Y$ be chosen randomly and $z_{ij} = f(y_{ij})$.

The **key K** consists of 2k y's and z's. y's form the secret key, z's form the public key.

Signing of a message $x = x_1 \dots x_k \in \{0,1\}^k$

$$sign(x_1 \dots x_k) = (y_{1,x_1}, \dots, y_{k,x_k}) = (a_1, \dots, a_k) \text{ - notation}$$

and

$$verif(x_1 \dots x_k, a_1, \dots, a_k) = true \Leftrightarrow f(a_i) = z_{i,xi}, 1 \leq i \leq k$$

Eve cannot forge a signature because she is unable to invert one-way functions.

Important note: Lamport signature scheme can be used safely to sign only one message. Why?

MERKLE SIGNATURES - I.

MERKLE SIGNATURES - I.

Merkle signature scheme with a parameter $m = 2^n$ allows to sign any of the given 2^n messages (and no other).

Merkle signature scheme with a parameter $m = 2^n$ allows to sign any of the given 2^n messages (and no other).

The scheme is based on so-called **hash trees** and uses **a fixed collision resistant hash function** h as well as **Lamport one-time signatures** and its security depends on their security.

Merkle signature scheme with a parameter $m = 2^n$ allows to sign any of the given 2^n messages (and no other).

The scheme is based on so-called **hash trees** and uses **a fixed collision resistant hash function** h as well as **Lamport one-time signatures** and its security depends on their security.

The main reason why Merkle Signature Scheme is of interest, is that it is believed to be resistant to potential attacks using quantum computers.

WILL WE HAVE (QUITE SOON) QUANTUM COMPUTERS?

- Who knows.

WILL WE HAVE (QUITE SOON) QUANTUM COMPUTERS?

- Who knows.
- The possibility of having quite soon powerful quantum computers starts to be so realistic that in US decision has been made, on a very-high level of cares for national security, that the next generation of cryptographic primitives' standards (for encryptions, digital signatures, hash functions,...) should be secure even in case quantum computers would be available.

MERKLE SIGNATURES - II.

Public key generation - a single key for all signings.

MERKLE SIGNATURES - II.

Public key generation - a single key for all signings. At first one needs to generate public keys PK_i and secret keys SK_i for all 2^n messages m_i, using Lamport signature scheme, and to compute also $h(PK_i)$ for all $i \leq 2^n$.

MERKLE SIGNATURES - II.

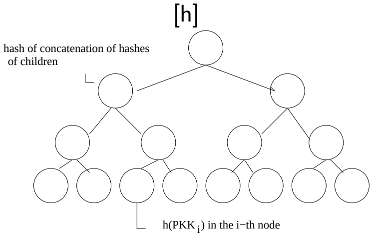
Public key generation - a single key for all signings. At first one needs to generate public keys PK_i and secret keys SK_i for all 2^n messages m_i, using Lamport signature scheme, and to compute also $h(PK_i)$ for all $i \leq 2^n$.

As the next step a complete binary tree with 2^n leaves is designed and the value $h(PK_i)$ is stored in the i-the leave, counting from left to right. Moreover, to each internal node the hash of the concatenation of hashes of its two children is stored.

MERKLE SIGNATURES - II.

Public key generation - a single key for all signings. At first one needs to generate public keys PK_i and secret keys SK_i for all 2^n messages m_i, using Lamport signature scheme, and to compute also $h(PK_i)$ for all $i \leq 2^n$.

As the next step a complete binary tree with 2^n leaves is designed and the value $h(PK_i)$ is stored in the i-the leave, counting from left to right. Moreover, to each internal node the hash of the concatenation of hashes of its two children is stored. The hash assigned this way to the root is the public key of the Merkle signature scheme and the tree is called **Merkle tree**. See next figure for a Merkle tree.



MERKLE SIGNATURE - III.

Signature generation. To sign a message m_i, this message is at first signed using the one-use signature scheme with keys (PK_i, SK_i).

MERKLE SIGNATURE - III.

Signature generation. To sign a message m_i, this message is at first signed using the one-use signature scheme with keys (PK_i, SK_i). **This signature plus a sequence of n hashes chosen from all those nodes that are needed to compute the hash of the root, is the Merkle signature**.

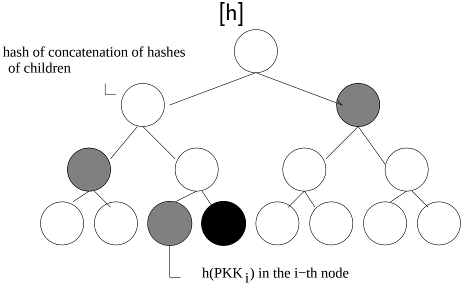
MERKLE SIGNATURE - III.

Signature generation. To sign a message m_i, this message is at first signed using the one-use signature scheme with keys (PK_i, SK_i). **This signature plus a sequence of n hashes chosen from all those nodes that are needed to compute the hash of the root, is the Merkle signature**. See the next Figure where the one-use sinature inhe black node and a sequence of gray nodes form the final signature.

MERKLE SIGNATURE - III.

Signature generation. To sign a message m_i, this message is at first signed using the one-use signature scheme with keys (PK_i, SK_i). **This signature plus a sequence of n hashes chosen from all those nodes that are needed to compute the hash of the root, is the Merkle signature**. See the next Figure where the one-use sinature inhe black node and a sequence of gray nodes form the final signature.

The verifier knows the public key - hash assigned to the root and signature created as above. This allows him to compute all hashes assigned to the root from the leave to the root and to verify that the value assigned this way agrees with he public key - hash assigned to the root.



GMR SIGNATURE SCHEME

In 1988 Shafi Goldwasser, Silvio Micali and Ronald Rivest were the first to define rigorously security requirements for digital signature schemes.

In 1988 Shafi Goldwasser, Silvio Micali and Ronald Rivest were the first to define rigorously security requirements for digital signature schemes.

They also presented a new signature scheme, known nowadays as **GMR signature scheme**.

GMR SIGNATURE SCHEME

In 1988 Shafi Goldwasser, Silvio Micali and Ronald Rivest were the first to define rigorously security requirements for digital signature schemes.
They also presented a new signature scheme, known nowadays as **GMR signature scheme**.

It was the first signature scheme that was proven as being robust against an adaptive chosen message attacks:

In 1988 Shafi Goldwasser, Silvio Micali and Ronald Rivest were the first to define rigorously security requirements for digital signature schemes.
They also presented a new signature scheme, known nowadays as **GMR signature scheme**.

It was the first signature scheme that was proven as being robust against an adaptive chosen message attacks: an adversary who receives signatures of messages of his choice (where each message may be chosen in a way that depends on the signatures of previously chosen messages) cannot later forge the signature even of a single additional message.

TIMESTAMPING

There are various ways that a digital signature can be compromised.

For example: if Eve determines the secret key of Bob, then she can forge signatures of any Bob's message she likes. If this happens, authenticity of all messages signed by Bob before Eve got the secret key is to be questioned.

The key problem is that there is no way to determine when a message was signed.

A timestamping protocol should provide a proof that a message was signed at a certain time.

TIMESTAMPING

There are various ways that a digital signature can be compromised.

For example: if Eve determines the secret key of Bob, then she can forge signatures of any Bob's message she likes. If this happens, authenticity of all messages signed by Bob before Eve got the secret key is to be questioned.

The key problem is that there is no way to determine when a message was signed.

A timestamping protocol should provide a proof that a message was signed at a certain time.

In the following pub denotes some publicly known information that could not be predicted before the day of the signature (for example, stock-market data).

TIMESTAMPING

There are various ways that a digital signature can be compromised.

For example: if Eve determines the secret key of Bob, then she can forge signatures of any Bob's message she likes. If this happens, authenticity of all messages signed by Bob before Eve got the secret key is to be questioned.

The key problem is that there is no way to determine when a message was signed.

A timestamping protocol should provide a proof that a message was signed at a certain time.

In the following pub denotes some publicly known information that could not be predicted before the day of the signature (for example, stock-market data).

Timestamping by Bob of a signature on a message w, using a hash function h.

- Bob computes z = h(w);
- Bob computes z' = h(z ∥ pub); – { ∥} denotes concatenation
- Bob computes y = sig(z');
- Bob publishes (z, pub, y) in the next day newspaper.

It is now clear that signature could not be done after the triple (z, pub, y) was published, but also not before the date pub was known.

BLIND SIGNATURES

The problem is whether Alice can make Bob to sign a message, say m, without Bob knowing m, therefore blindly.

BLIND SIGNATURES

The problem is whether Alice can make Bob to sign a message, say m, without Bob knowing m, therefore blindly.

– this would be needed, for example, in e-commerce.

BLIND SIGNATURES

The problem is whether Alice can make Bob to sign a message, say m, without Bob knowing m, therefore blindly.

– this would be needed, for example, in e-commerce.

She can. Blind signing can be realized by a two party protocol, between the Alice and Bob, that has the following properties.

BLIND SIGNATURES

The problem is whether Alice can make Bob to sign a message, say m, without Bob knowing m, therefore blindly.

– this would be needed, for example, in e-commerce.

She can. Blind signing can be realized by a two party protocol, between the Alice and Bob, that has the following properties.

- In order to sign (by Bob) a message m, Alice creates, using a blinding procedure, from the message m a new message $m*$ from which m can not be obtained without knowing a secret, and sends $m*$ to Bob for signing.

BLIND SIGNATURES

The problem is whether Alice can make Bob to sign a message, say m, without Bob knowing m, therefore blindly.

– this would be needed, for example, in e-commerce.

She can. Blind signing can be realized by a two party protocol, between the Alice and Bob, that has the following properties.

- In order to sign (by Bob) a message m, Alice creates, using a blinding procedure, from the message m a new message $m*$ from which m can not be obtained without knowing a secret, and sends $m*$ to Bob for signing.
- Bob signs the message $m*$ to get a signature s_{m^*} (of $m*$) and sends s_{m*} to Alice.

BLIND SIGNATURES

The problem is whether Alice can make Bob to sign a message, say m, without Bob knowing m, therefore blindly.

– this would be needed, for example, in e-commerce.

She can. Blind signing can be realized by a two party protocol, between the Alice and Bob, that has the following properties.

- In order to sign (by Bob) a message m, Alice creates, using a blinding procedure, from the message m a new message $m*$ from which m can not be obtained without knowing a secret, and sends $m*$ to Bob for signing.
- Bob signs the message $m*$ to get a signature s_{m^*} (of $m*$) and sends s_{m*} to Alice. The signing is to be done in such a way that Alice can afterwards compute, using an unblinding procedure, from Bob's signature s_{m*} of $m*$ – Bob's signature s_m of m.

Chaum's BLIND SIGNATURE SCHEME

This blind signature protocol combines RSA with blinding/unblinding features.

Chaum's BLIND SIGNATURE SCHEME

This blind signature protocol combines RSA with blinding/unblinding features.

Let Bob's RSA public key be (n, e) and his private key be d.
Let m be a message, $0 < m < n$,

Chaum's BLIND SIGNATURE SCHEME

This blind signature protocol combines RSA with blinding/unblinding features.

Let Bob's RSA public key be (n, e) and his private key be d.
Let m be a message, $0 < m < n$,

PROTOCOL:

- Alice chooses a random $0 < k < n$ with $\gcd(n, k) = 1$.

Chaum's BLIND SIGNATURE SCHEME

This blind signature protocol combines RSA with blinding/unblinding features.

Let Bob's RSA public key be (n, e) and his private key be d.
Let m be a message, $0 < m < n$,

PROTOCOL:

- Alice chooses a random $0 < k < n$ with $\gcd(n, k) = 1$.
- Alice computes $m^* = mk^e \pmod{n}$ and sends it to Bob (this way Alice blinds the message m).

Chaum's BLIND SIGNATURE SCHEME

This blind signature protocol combines RSA with blinding/unblinding features.

Let Bob's RSA public key be (n, e) and his private key be d.
Let m be a message, $0 < m < n$,

PROTOCOL:

- Alice chooses a random $0 < k < n$ with $\gcd(n, k) = 1$.
- Alice computes $m^* = mk^e \pmod{n}$ and sends it to Bob (this way Alice blinds the message m).
- Bob computed $s^* = (m^*)^d \pmod{n}$ and sends s* to Alice (this way Bob signs the blinded message m*).

Chaum's BLIND SIGNATURE SCHEME

This blind signature protocol combines RSA with blinding/unblinding features.

Let Bob's RSA public key be (n, e) and his private key be d.
Let m be a message, $0 < m < n$,

PROTOCOL:

- Alice chooses a random $0 < k < n$ with $\gcd(n, k) = 1$.
- Alice computes $m^* = mk^e \pmod{n}$ and sends it to Bob (this way Alice blinds the message m).
- Bob computed $s^* = (m^*)^d \pmod{n}$ and sends s* to Alice (this way Bob signs the blinded message m*).
- Alice computes $s = k^{-1}s^* \pmod{n}$ to obtain Bob's signature m^d of m (This way Alice performs unblinding of m^*).

Chaum's BLIND SIGNATURE SCHEME

This blind signature protocol combines RSA with blinding/unblinding features.

Let Bob's RSA public key be (n, e) and his private key be d.
Let m be a message, $0 < m < n$,

PROTOCOL:

- Alice chooses a random $0 < k < n$ with $\gcd(n, k) = 1$.
- Alice computes $m^* = mk^e$ (**mod** n) and sends it to Bob (this way Alice blinds the message m).
- Bob computed $s^* = (m^*)^d$ (**mod** n) and sends s* to Alice (this way Bob signs the blinded message m*).
- Alice computes $s = k^{-1}s^*$(**mod** n) to obtain Bob's signature m^d of m (This way Alice performs unblinding of m^*).

Verification is similar to that of the RSA signature scheme.

DIGITAL SIGNATURES with ENCRYPTION and RESENDING

Let us consider the following communication between Alice and Bob:

DIGITAL SIGNATURES with ENCRYPTION and RESENDING

Let us consider the following communication between Alice and Bob:

1. Alice signs the message: $s_A(w)$.

DIGITAL SIGNATURES with ENCRYPTION and RESENDING

Let us consider the following communication between Alice and Bob:

1. Alice signs the message: $s_A(w)$.
2. Alice encrypts the signed message: $e_B(s_A(w))$ and sends it to Bob.

DIGITAL SIGNATURES with ENCRYPTION and RESENDING

Let us consider the following communication between Alice and Bob:

1. Alice signs the message: $s_A(w)$.
2. Alice encrypts the signed message: $e_B(s_A(w))$ and sends it to Bob.
3. Bob decrypts the signed message: $d_B(e_B(s_A(w)))$

DIGITAL SIGNATURES with ENCRYPTION and RESENDING

Let us consider the following communication between Alice and Bob:

1. Alice signs the message: $s_A(w)$.
2. Alice encrypts the signed message: $e_B(s_A(w))$ and sends it to Bob.
3. Bob decrypts the signed message: $d_B(e_B(s_A(w))) = s_A(w)$.
4. Bob verifies the signature and recovers the message $v_A(s_A(w)) = w$.

DIGITAL SIGNATURES with ENCRYPTION and RESENDING

Let us consider the following communication between Alice and Bob:

1. Alice signs the message: $s_A(w)$.
2. Alice encrypts the signed message: $e_B(s_A(w))$ and sends it to Bob.
3. Bob decrypts the signed message: $d_B(e_B(s_A(w))) = s_A(w)$.
4. Bob verifies the signature and recovers the message $v_A(s_A(w)) = w$.

Consider now the case of resending the message as a receipt

5. Bob signs and encrypts the message and sends to Alice $e_A(s_B(w))$.

DIGITAL SIGNATURES with ENCRYPTION and RESENDING

Let us consider the following communication between Alice and Bob:

1. Alice signs the message: $s_A(w)$.
2. Alice encrypts the signed message: $e_B(s_A(w))$ and sends it to Bob.
3. Bob decrypts the signed message: $d_B(e_B(s_A(w))) = s_A(w)$.
4. Bob verifies the signature and recovers the message $v_A(s_A(w)) = w$.

Consider now the case of resending the message as a receipt

5. Bob signs and encrypts the message and sends to Alice $e_A(s_B(w))$.
6. Alice decrypts the message and verifies the signature.

DIGITAL SIGNATURES with ENCRYPTION and RESENDING

Let us consider the following communication between Alice and Bob:

1. Alice signs the message: $s_A(w)$.
2. Alice encrypts the signed message: $e_B(s_A(w))$ and sends it to Bob.
3. Bob decrypts the signed message: $d_B(e_B(s_A(w))) = s_A(w)$.
4. Bob verifies the signature and recovers the message $v_A(s_A(w)) = w$.

Consider now the case of resending the message as a receipt

5. Bob signs and encrypts the message and sends to Alice $e_A(s_B(w))$.
6. Alice decrypts the message and verifies the signature.

Assume now: $v_x = e_x$, $s_x = d_x$ for all users x.

A SURPRISING ATTACK to the PREVIOUS SCHEME

1. Mallot intercepts $e_B(s_A(w))$.

A SURPRISING ATTACK to the PREVIOUS SCHEME

1. Mallot intercepts $e_B(s_A(w))$.
2. Later Mallot sends $e_B(s_A(w))$ to Bob pretending it is from him (from Mallot).

A SURPRISING ATTACK to the PREVIOUS SCHEME

1. Mallot intercepts $e_B(s_A(w))$.
2. Later Mallot sends $e_B(s_A(w))$ to Bob pretending it is from him (from Mallot).
3. Bob decrypts and "verifies" the message by computing

$$e_M(s_B(e_B(s_A(w)))) = e_M(s_A(w)) \text{ – a garbage.}$$

A SURPRISING ATTACK to the PREVIOUS SCHEME

1. Mallot intercepts $e_B(s_A(w))$.
2. Later Mallot sends $e_B(s_A(w))$ to Bob pretending it is from him (from Mallot).
3. Bob decrypts and "verifies" the message by computing
   $$e_M(s_B(e_B(s_A(w)))) = e_M(s_A(w)) \text{ – a garbage.}$$
4. Bob goes on with the protocol and returns to Mallot the receipt:
   $$e_M(s_B(e_M(s_A(w))))$$

A SURPRISING ATTACK to the PREVIOUS SCHEME

1. Mallot intercepts $e_B(s_A(w))$.
2. Later Mallot sends $e_B(s_A(w))$ to Bob pretending it is from him (from Mallot).
3. Bob decrypts and "verifies" the message by computing

   $e_M(s_B(e_B(s_A(w)))) = e_M(s_A(w))$ – a garbage.
4. Bob goes on with the protocol and returns to Mallot the receipt:

   $$e_M(s_B(e_M(s_A(w))))$$
5. Mallot can then get w (observe that $v_X = e_X$ and $s_x = d_x$ for each user x).

   Indeed, Mallot can compute

   $e_A(s_M(e_B(s_M(e_M(s_B(e_M(s_A(w)))))))) =$ w.

ANOTHER MAN-IN-THE-MIDDLE ATTACK

Consider the following protocol:

1. Alice sends the pair $(e_B(e_B(w)||A), B)$ to Bob.
2. Bob uses d_B to get A and w, and acknowledges the receipt by sending the pair $(e_A(e_A(w)||B), A)$ to Alice.

(Here the function e and d are assumed to operate on strings and identificators $A, B, \ldots$ are strings.)

ANOTHER MAN-IN-THE-MIDDLE ATTACK

Consider the following protocol:

1. Alice sends the pair $(e_B(e_B(w)||A), B)$ to Bob.
2. Bob uses d_B to get A and w, and acknowledges the receipt by sending the pair $(e_A(e_A(w)||B), A)$ to Alice.

(Here the function e and d are assumed to operate on strings and identificators $A, B, \dots$ are strings.)

What can an active eavesdropper C do?

- C can learn $(e_A(e_A(w)||B), A)$ and therefore $e_A(w')$ for $w' = e_A(w)||B$.

ANOTHER MAN-IN-THE-MIDDLE ATTACK

Consider the following protocol:

1. Alice sends the pair $(e_B(e_B(w)||A), B)$ to Bob.
2. Bob uses d_B to get A and w, and acknowledges the receipt by sending the pair $(e_A(e_A(w)||B), A)$ to Alice.

(Here the function e and d are assumed to operate on strings and identificators $A, B, \ldots$ are strings.)

What can an active eavesdropper C do?

- C can learn $(e_A(e_A(w)||B), A)$ and therefore $e_A(w')$ for $w' = e_A(w)||B$.
- C can now send to Alice the pair $(e_A(e_A||w')||C), A)$.

ANOTHER MAN-IN-THE-MIDDLE ATTACK

Consider the following protocol:

1. Alice sends the pair $(e_B(e_B(w)||A), B)$ to Bob.
2. Bob uses d_B to get A and w, and acknowledges the receipt by sending the pair $(e_A(e_A(w)||B), A)$ to Alice.

(Here the function e and d are assumed to operate on strings and identificators $A, B, \ldots$ are strings.)

What can an active eavesdropper C do?

- C can learn $(e_A(e_A(w)||B), A)$ and therefore $e_A(w')$ for $w' = e_A(w)||B$.
- C can now send to Alice the pair $(e_A(e_A||w')||C), A)$.
- Alice, thinking that this is the step 1 of the protocol, acknowledges the receipt by sending the pair $(e_C(e_C(w')||A), C)$ to C.

ANOTHER MAN-IN-THE-MIDDLE ATTACK

Consider the following protocol:

1. Alice sends the pair $(e_B(e_B(w)||A), B)$ to Bob.
2. Bob uses d_B to get A and w, and acknowledges the receipt by sending the pair $(e_A(e_A(w)||B), A)$ to Alice.

(Here the function e and d are assumed to operate on strings and identificators $A, B, \ldots$ are strings.)

What can an active eavesdropper C do?

- C can learn $(e_A(e_A(w)||B), A)$ and therefore $e_A(w')$ for $w' = e_A(w)||B$.
- C can now send to Alice the pair $(e_A(e_A||w')||C), A)$.
- Alice, thinking that this is the step 1 of the protocol, acknowledges the receipt by sending the pair $(e_C(e_C(w')||A), C)$ to C.
- C is now able to learn w' and therefore also $e_A(w)$.
- C now sends to Alice the pair $(e_A(e_A(w)||C), A)$.

ANOTHER MAN-IN-THE-MIDDLE ATTACK

Consider the following protocol:

1. Alice sends the pair $(e_B(e_B(w)||A), B)$ to Bob.
2. Bob uses d_B to get A and w, and acknowledges the receipt by sending the pair $(e_A(e_A(w)||B), A)$ to Alice.

(Here the function e and d are assumed to operate on strings and identificators $A, B, \ldots$ are strings.)

What can an active eavesdropper C do?

- C can learn $(e_A(e_A(w)||B), A)$ and therefore $e_A(w')$ for $w' = e_A(w)||B$.
- C can now send to Alice the pair $(e_A(e_A||w')||C), A)$.
- Alice, thinking that this is the step 1 of the protocol, acknowledges the receipt by sending the pair $(e_C(e_C(w')||A), C)$ to C.
- C is now able to learn w' and therefore also $e_A(w)$.
- C now sends to Alice the pair $(e_A(e_A(w)||C), A)$.
- Alice makes acknowledgment by sending the pair $(e_C(e_C(w)||A), C)$.
- C is now able to learn w.

PROBABILISTIC SIGNATURES SCHEMES - PSS

Let us have **integers** k, l, n such that $k + l < n$, a **trapdoor permutation**

$$f : D \to D, D \subset \{0,1\}^n,$$

a **pseudorandom bit generator**

$$G : \{0,1\}^l \to \{0,1\}^k \times \{0,1\}^{n-(l+k)}, \quad G(w) = (G_1(w), G_2(w))$$

and a **hash function**

$$h : \{0,1\}^* \to \{0,1\}^l.$$

The following PSS scheme is applicable to messages of arbitrary length.

PROBABILISTIC SIGNATURES SCHEMES - PSS

Let us have **integers** k, l, n such that $k + l < n$, a **trapdoor permutation**

$$f : D \to D, D \subset \{0,1\}^n,$$

a **pseudorandom bit generator**

$$G : \{0,1\}^l \to \{0,1\}^k \times \{0,1\}^{n-(l+k)}, \quad G(w) = (G_1(w), G_2(w))$$

and a **hash function**

$$h : \{0,1\}^* \to \{0,1\}^l.$$

The following PSS scheme is applicable to messages of arbitrary length.

Signing: of a message $w \in \{0,1\}^*$.

1. Choose random $r \in \{0,1\}^k$ and compute $m = h(w\|r)$.
2. Compute $G(m) = (G_1(m), G_2(m))$ and $y = m\|(G_1(m) \oplus r)\|G_2(m)$.
3. Signature of w is $\sigma = f^{-1}(y)$.

PROBABILISTIC SIGNATURES SCHEMES - PSS

Let us have **integers** k, l, n such that $k + l < n$, a **trapdoor permutation**

$$f : D \to D, D \subset \{0,1\}^n,$$

a **pseudorandom bit generator**

$$G : \{0,1\}^l \to \{0,1\}^k \times \{0,1\}^{n-(l+k)}, \quad G(w) = (G_1(w), G_2(w))$$

and a **hash function**

$$h : \{0,1\}^* \to \{0,1\}^l.$$

The following PSS scheme is applicable to messages of arbitrary length.

Signing: of a message $w \in \{0,1\}^*$.

1. Choose random $r \in \{0,1\}^k$ and compute $m = h(w\|r)$.
2. Compute $G(m) = (G_1(m), G_2(m))$ and $y = m\|(G_1(m) \oplus r)\|G_2(m)$.
3. Signature of w is $\sigma = f^{-1}(y)$.

Verification of a signed message (w, σ).

PROBABILISTIC SIGNATURES SCHEMES - PSS

Let us have **integers** k, l, n such that $k + l < n$, a **trapdoor permutation**

$$f : D \to D, D \subset \{0,1\}^n,$$

a **pseudorandom bit generator**

$$G : \{0,1\}^l \to \{0,1\}^k \times \{0,1\}^{n-(l+k)}, \quad G(w) = (G_1(w), G_2(w))$$

and a **hash function**

$$h : \{0,1\}^* \to \{0,1\}^l.$$

The following PSS scheme is applicable to messages of arbitrary length.

Signing: of a message $w \in \{0,1\}^*$.

1. Choose random $r \in \{0,1\}^k$ and compute $m = h(w\|r)$.
2. Compute $G(m) = (G_1(m), G_2(m))$ and $y = m\|(G_1(m) \oplus r)\|G_2(m)$.
3. Signature of w is $\sigma = f^{-1}(y)$.

Verification of a signed message (w, σ).

- Compute $f(\sigma)$ and decompose $f(\sigma) = m\|t\|u$, where $|m| = l$, $|t| = k$ and $|u| = n - (k + l)$.

PROBABILISTIC SIGNATURES SCHEMES - PSS

Let us have **integers** k, l, n such that $k + l < n$, a **trapdoor permutation**

$$f : D \to D, D \subset \{0,1\}^n,$$

a **pseudorandom bit generator**

$$G : \{0,1\}^l \to \{0,1\}^k \times \{0,1\}^{n-(l+k)}, \quad G(w) = (G_1(w), G_2(w))$$

and a **hash function**

$$h : \{0,1\}^* \to \{0,1\}^l.$$

The following PSS scheme is applicable to messages of arbitrary length.

Signing: of a message $w \in \{0,1\}^*$.

1. Choose random $r \in \{0,1\}^k$ and compute $m = h(w\|r)$.
2. Compute $G(m) = (G_1(m), G_2(m))$ and $y = m\|(G_1(m) \oplus r)\|G_2(m)$.
3. Signature of w is $\sigma = f^{-1}(y)$.

Verification of a signed message (w, σ).

- Compute $f(\sigma)$ and decompose $f(\sigma) = m\|t\|u$, where $|m| = l$, $|t| = k$ and $|u| = n - (k + l)$.
- Compute $r = t \oplus G_1(m)$.

PROBABILISTIC SIGNATURES SCHEMES - PSS

Let us have **integers** k, l, n such that $k + l < n$, a **trapdoor permutation**

$$f : D \to D, D \subset \{0,1\}^n,$$

a **pseudorandom bit generator**

$$G : \{0,1\}^l \to \{0,1\}^k \times \{0,1\}^{n-(l+k)}, \quad G(w) = (G_1(w), G_2(w))$$

and a **hash function**

$$h : \{0,1\}^* \to \{0,1\}^l.$$

The following PSS scheme is applicable to messages of arbitrary length.

Signing: of a message $w \in \{0,1\}^*$.

1. Choose random $r \in \{0,1\}^k$ and compute $m = h(w\|r)$.
2. Compute $G(m) = (G_1(m), G_2(m))$ and $y = m\|(G_1(m) \oplus r)\|G_2(m)$.
3. Signature of w is $\sigma = f^{-1}(y)$.

Verification of a signed message (w, σ).

- Compute $f(\sigma)$ and decompose $f(\sigma) = m\|t\|u$, where $|m| = l$, $|t| = k$ and $|u| = n - (k + l)$.
- Compute $r = t \oplus G_1(m)$.
- Accept signature σ if $h(w\|r) = m$ and $G_2(m) = u$; otherwise reject it.

Diffie-Hellman PUBLIC ESTABLISHMENT of SECRET KEYS - repetition

Main problem of the secret-key cryptography: a need to make a secure distribution (establishment) of secret keys ahead of transmissions.

Diffie+Hellman solved this problem in 1976 by designing a protocol for secure key establishment (distribution) over public channels.

Diffie-Hellman PUBLIC ESTABLISHMENT of SECRET KEYS - repetition

Main problem of the secret-key cryptography: a need to make a secure distribution (establishment) of secret keys ahead of transmissions.

Diffie+Hellman solved this problem in 1976 by designing a protocol for secure key establishment (distribution) over public channels.

Diffie-Hellman Protocol: If two parties, Alice and Bob, want to create a common secret key, then they first agree, somehow, on a large prime p and a q<p of large order in Z_p^* and then they perform, through a public channel, the following activities.

- Alice chooses, randomly, a large $1 \leq x < p - 1$ and computes

$$X = q^x \bmod p.$$

Diffie-Hellman PUBLIC ESTABLISHMENT of SECRET KEYS - repetition

Main problem of the secret-key cryptography: a need to make a secure distribution (establishment) of secret keys ahead of transmissions.

Diffie+Hellman solved this problem in 1976 by designing a protocol for secure key establishment (distribution) over public channels.

Diffie-Hellman Protocol: If two parties, Alice and Bob, want to create a common secret key, then they first agree, somehow, on a large prime p and a q<p of large order in Z_p^* and then they perform, through a public channel, the following activities.

- Alice chooses, randomly, a large $1 \leq x < p - 1$ and computes
$$X = q^x \bmod p.$$
- Bob also chooses, again randomly, a large $1 \leq y < p - 1$ and computes
$$Y = q^y \bmod p.$$

Diffie-Hellman PUBLIC ESTABLISHMENT of SECRET KEYS - repetition

Main problem of the secret-key cryptography: a need to make a secure distribution (establishment) of secret keys ahead of transmissions.

Diffie+Hellman solved this problem in 1976 by designing a protocol for secure key establishment (distribution) over public channels.

Diffie-Hellman Protocol: If two parties, Alice and Bob, want to create a common secret key, then they first agree, somehow, on a large prime p and a q<p of large order in Z_p^* and then they perform, through a public channel, the following activities.

- Alice chooses, randomly, a large $1 \leq x < p-1$ and computes
$$X = q^x \bmod p.$$
- Bob also chooses, again randomly, a large $1 \leq y < p-1$ and computes
$$Y = q^y \bmod p.$$
- Alice and Bob exchange **X** and **Y**, through a public channel, but keep **x, y** secret.

Diffie-Hellman PUBLIC ESTABLISHMENT of SECRET KEYS - repetition

Main problem of the secret-key cryptography: a need to make a secure distribution (establishment) of secret keys ahead of transmissions.

Diffie+Hellman solved this problem in 1976 by designing a protocol for secure key establishment (distribution) over public channels.

Diffie-Hellman Protocol: If two parties, Alice and Bob, want to create a common secret key, then they first agree, somehow, on a large prime p and a q<p of large order in Z_p^* and then they perform, through a public channel, the following activities.

- Alice chooses, randomly, a large $1 \leq x < p-1$ and computes
$$X = q^x \bmod p.$$
- Bob also chooses, again randomly, a large $1 \leq y < p-1$ and computes
$$Y = q^y \bmod p.$$
- Alice and Bob exchange **X** and **Y**, through a public channel, but keep **x, y** secret.
- Alice computes $Y^x \bmod p$ and Bob computes $X^y \bmod p$ and then each of them has the key
$$K = q^{xy} \bmod p.$$

Diffie-Hellman PUBLIC ESTABLISHMENT of SECRET KEYS - repetition

Main problem of the secret-key cryptography: a need to make a secure distribution (establishment) of secret keys ahead of transmissions.

Diffie+Hellman solved this problem in 1976 by designing a protocol for secure key establishment (distribution) over public channels.

Diffie-Hellman Protocol: If two parties, Alice and Bob, want to create a common secret key, then they first agree, somehow, on a large prime p and a q<p of large order in Z_p^* and then they perform, through a public channel, the following activities.

- Alice chooses, randomly, a large $1 \leq x < p-1$ and computes
$$X = q^x \bmod p.$$
- Bob also chooses, again randomly, a large $1 \leq y < p-1$ and computes
$$Y = q^y \bmod p.$$
- Alice and Bob exchange **X** and **Y**, through a public channel, but keep **x, y** secret.
- Alice computes $Y^x \bmod p$ and Bob computes $X^y \bmod p$ and then each of them has the key
$$K = q^{xy} \bmod p.$$

Diffie-Hellman PUBLIC ESTABLISHMENT of SECRET KEYS - repetition

Main problem of the secret-key cryptography: a need to make a secure distribution (establishment) of secret keys ahead of transmissions.

Diffie+Hellman solved this problem in 1976 by designing a protocol for secure key establishment (distribution) over public channels.

Diffie-Hellman Protocol: If two parties, Alice and Bob, want to create a common secret key, then they first agree, somehow, on a large prime p and a q<p of large order in Z_p^* and then they perform, through a public channel, the following activities.

- Alice chooses, randomly, a large $1 \leq x < p-1$ and computes
$$X = q^x \bmod p.$$
- Bob also chooses, again randomly, a large $1 \leq y < p-1$ and computes
$$Y = q^y \bmod p.$$
- Alice and Bob exchange **X** and **Y**, through a public channel, but keep **x, y** secret.
- Alice computes $Y^x \bmod p$ and Bob computes $X^y \bmod p$ and then each of them has the key
$$K = q^{xy} \bmod p.$$

An eavesdropper seems to need, in order to determine x from **X, q, p** and y from **Y, q, p**, a capability to compute discrete logarithms, or to compute q^{xy} from q^x and q^y, what is believed to be infeasible.

AUTHENTICATED Diffie-Hellman KEY EXCHANGE

AUTHENTICATED Diffie-Hellman KEY EXCHANGE

Let each user U have a signature algorithm s_U and a verification algorithm v_U.

AUTHENTICATED Diffie-Hellman KEY EXCHANGE

Let each user U have a signature algorithm s_U and a verification algorithm v_U.

The following protocol allows Alice and Bob to establish a key K to use with an encryption function e_K and to avoid the man-in-the-middle attack.

1. Alice and Bob choose large prime p and a generator $q \in Z_p^*$.

AUTHENTICATED Diffie-Hellman KEY EXCHANGE

Let each user U have a signature algorithm s_U and a verification algorithm v_U.
The following protocol allows Alice and Bob to establish a key K to use with an encryption function e_K and to avoid the man-in-the-middle attack.

1. Alice and Bob choose large prime p and a generator $q \in Z_p^*$.
2. Alice chooses a random x and Bob chooses a random y.

AUTHENTICATED Diffie-Hellman KEY EXCHANGE

Let each user U have a signature algorithm s_U and a verification algorithm v_U.
The following protocol allows Alice and Bob to establish a key K to use with an encryption function e_K and to avoid the man-in-the-middle attack.

1. Alice and Bob choose large prime p and a generator $q \in Z_p^*$.
2. Alice chooses a random x and Bob chooses a random y.
3. Alice computes $q^x \mod p$, and Bob computes $q^y \mod p$.

AUTHENTICATED Diffie-Hellman KEY EXCHANGE

Let each user U have a signature algorithm s_U and a verification algorithm v_U.
The following protocol allows Alice and Bob to establish a key K to use with an encryption function e_K and to avoid the man-in-the-middle attack.

1. Alice and Bob choose large prime p and a generator $q \in Z_p^*$.
2. Alice chooses a random x and Bob chooses a random y.
3. Alice computes $q^x \mod p$, and Bob computes $q^y \mod p$.
4. Alice sends q^x to Bob.

AUTHENTICATED Diffie-Hellman KEY EXCHANGE

Let each user U have a signature algorithm s_U and a verification algorithm v_U.
The following protocol allows Alice and Bob to establish a key K to use with an encryption function e_K and to avoid the man-in-the-middle attack.

1. Alice and Bob choose large prime p and a generator $q \in Z_p^*$.
2. Alice chooses a random x and Bob chooses a random y.
3. Alice computes $q^x \mod p$, and Bob computes $q^y \mod p$.
4. Alice sends q^x to Bob.
5. Bob computes $K = q^{xy} \mod p$.

AUTHENTICATED Diffie-Hellman KEY EXCHANGE

Let each user U have a signature algorithm s_U and a verification algorithm v_U.
The following protocol allows Alice and Bob to establish a key K to use with an encryption function e_K and to avoid the man-in-the-middle attack.

1. Alice and Bob choose large prime p and a generator $q \in Z_p^*$.
2. Alice chooses a random x and Bob chooses a random y.
3. Alice computes $q^x \mod p$, and Bob computes $q^y \mod p$.
4. Alice sends q^x to Bob.
5. Bob computes $K = q^{xy} \mod p$.
6. Bob sends q^y and $e_K(s_B(q^y, q^x))$ to Alice.

AUTHENTICATED Diffie-Hellman KEY EXCHANGE

Let each user U have a signature algorithm s_U and a verification algorithm v_U.
The following protocol allows Alice and Bob to establish a key K to use with an encryption function e_K and to avoid the man-in-the-middle attack.

1. Alice and Bob choose large prime p and a generator $q \in Z_p^*$.
2. Alice chooses a random x and Bob chooses a random y.
3. Alice computes $q^x \mod p$, and Bob computes $q^y \mod p$.
4. Alice sends q^x to Bob.
5. Bob computes $K = q^{xy} \mod p$.
6. Bob sends q^y and $e_K(s_B(q^y, q^x))$ to Alice.
7. Alice computes $K = q^{xy} \mod p$.

AUTHENTICATED Diffie-Hellman KEY EXCHANGE

Let each user U have a signature algorithm s_U and a verification algorithm v_U.
The following protocol allows Alice and Bob to establish a key K to use with an encryption function e_K and to avoid the man-in-the-middle attack.

1. Alice and Bob choose large prime p and a generator $q \in Z_p^*$.
2. Alice chooses a random x and Bob chooses a random y.
3. Alice computes $q^x \mod p$, and Bob computes $q^y \mod p$.
4. Alice sends q^x to Bob.
5. Bob computes $K = q^{xy} \mod p$.
6. Bob sends q^y and $e_K(s_B(q^y, q^x))$ to Alice.
7. Alice computes $K = q^{xy} \mod p$.
8. Alice decrypts $e_K(s_B(q^y, q^x))$ to obtain $s_B(q^y, q^x)$.

AUTHENTICATED Diffie-Hellman KEY EXCHANGE

Let each user U have a signature algorithm s_U and a verification algorithm v_U.
The following protocol allows Alice and Bob to establish a key K to use with an encryption function e_K and to avoid the man-in-the-middle attack.

1. Alice and Bob choose large prime p and a generator $q \in Z_p^*$.
2. Alice chooses a random x and Bob chooses a random y.
3. Alice computes $q^x \mod p$, and Bob computes $q^y \mod p$.
4. Alice sends q^x to Bob.
5. Bob computes $K = q^{xy} \mod p$.
6. Bob sends q^y and $e_K(s_B(q^y, q^x))$ to Alice.
7. Alice computes $K = q^{xy} \mod p$.
8. Alice decrypts $e_K(s_B(q^y, q^x))$ to obtain $s_B(q^y, q^x)$.
9. Alice gets, using an authority, Bob's verification algorithm v_B .

AUTHENTICATED Diffie-Hellman KEY EXCHANGE

Let each user U have a signature algorithm s_U and a verification algorithm v_U.
The following protocol allows Alice and Bob to establish a key K to use with an encryption function e_K and to avoid the man-in-the-middle attack.

1. Alice and Bob choose large prime p and a generator $q \in Z_p^*$.
2. Alice chooses a random x and Bob chooses a random y.
3. Alice computes $q^x \mod p$, and Bob computes $q^y \mod p$.
4. Alice sends q^x to Bob.
5. Bob computes $K = q^{xy} \mod p$.
6. Bob sends q^y and $e_K(s_B(q^y, q^x))$ to Alice.
7. Alice computes $K = q^{xy} \mod p$.
8. Alice decrypts $e_K(s_B(q^y, q^x))$ to obtain $s_B(q^y, q^x)$.
9. Alice gets, using an authority, Bob's verification algorithm v_B .
10. Alice uses v_B to verify Bob's signature.

AUTHENTICATED Diffie-Hellman KEY EXCHANGE

Let each user U have a signature algorithm s_U and a verification algorithm v_U.
The following protocol allows Alice and Bob to establish a key K to use with an encryption function e_K and to avoid the man-in-the-middle attack.

1. Alice and Bob choose large prime p and a generator $q \in Z_p^*$.
2. Alice chooses a random x and Bob chooses a random y.
3. Alice computes $q^x \mod p$, and Bob computes $q^y \mod p$.
4. Alice sends q^x to Bob.
5. Bob computes $K = q^{xy} \mod p$.
6. Bob sends q^y and $e_K(s_B(q^y, q^x))$ to Alice.
7. Alice computes $K = q^{xy} \mod p$.
8. Alice decrypts $e_K(s_B(q^y, q^x))$ to obtain $s_B(q^y, q^x)$.
9. Alice gets, using an authority, Bob's verification algorithm v_B .
10. Alice uses v_B to verify Bob's signature.
11. Alice sends $e_K(s_A(q^x, q^y))$ to Bob.

AUTHENTICATED Diffie-Hellman KEY EXCHANGE

Let each user U have a signature algorithm s_U and a verification algorithm v_U.
The following protocol allows Alice and Bob to establish a key K to use with an encryption function e_K and to avoid the man-in-the-middle attack.

1. Alice and Bob choose large prime p and a generator $q \in Z_p^*$.
2. Alice chooses a random x and Bob chooses a random y.
3. Alice computes $q^x \mod p$, and Bob computes $q^y \mod p$.
4. Alice sends q^x to Bob.
5. Bob computes $K = q^{xy} \mod p$.
6. Bob sends q^y and $e_K(s_B(q^y, q^x))$ to Alice.
7. Alice computes $K = q^{xy} \mod p$.
8. Alice decrypts $e_K(s_B(q^y, q^x))$ to obtain $s_B(q^y, q^x)$.
9. Alice gets, using an authority, Bob's verification algorithm v_B .
10. Alice uses v_B to verify Bob's signature.
11. Alice sends $e_K(s_A(q^x, q^y))$ to Bob.
12. Bob decrypts, gets v_A, and verifies Alice's signature.

AUTHENTICATED Diffie-Hellman KEY EXCHANGE

Let each user U have a signature algorithm s_U and a verification algorithm v_U.
The following protocol allows Alice and Bob to establish a key K to use with an encryption function e_K and to avoid the man-in-the-middle attack.

1. Alice and Bob choose large prime p and a generator $q \in Z_p^*$.
2. Alice chooses a random x and Bob chooses a random y.
3. Alice computes $q^x \mod p$, and Bob computes $q^y \mod p$.
4. Alice sends q^x to Bob.
5. Bob computes $K = q^{xy} \mod p$.
6. Bob sends q^y and $e_K(s_B(q^y, q^x))$ to Alice.
7. Alice computes $K = q^{xy} \mod p$.
8. Alice decrypts $e_K(s_B(q^y, q^x))$ to obtain $s_B(q^y, q^x)$.
9. Alice gets, using an authority, Bob's verification algorithm v_B .
10. Alice uses v_B to verify Bob's signature.
11. Alice sends $e_K(s_A(q^x, q^y))$ to Bob.
12. Bob decrypts, gets v_A, and verifies Alice's signature.

An enhanced version of the above protocol is known as Station-to-Station protocol.

THRESHOLD DIGITAL SIGNATURES

The idea of a (t+1, n) threshold signature scheme is to distribute the power of the signing operation to (t+1) parties out of n.

THRESHOLD DIGITAL SIGNATURES

The idea of a (t+1, n) threshold signature scheme is to distribute the power of the signing operation to (t+1) parties out of n.

A (t+1) threshold signature scheme should satisfy two conditions.

THRESHOLD DIGITAL SIGNATURES

The idea of a (t+1, n) threshold signature scheme is to distribute the power of the signing operation to (t+1) parties out of n.

A (t+1) threshold signature scheme should satisfy two conditions.

Unforgeability means that even if an adversary corrupts t parties, he still cannot generate a valid signature.

THRESHOLD DIGITAL SIGNATURES

The idea of a (t+1, n) threshold signature scheme is to distribute the power of the signing operation to (t+1) parties out of n.

A (t+1) threshold signature scheme should satisfy two conditions.

Unforgeability means that even if an adversary corrupts t parties, he still cannot generate a valid signature.

Robustness means that corrupted parties cannot prevent uncorrupted parties to generate signatures.

Shoup (2000) presented an efficient, non-interactive, robust and unforgeable threshold RSA signature schemes.

There is no proof yet whether Shoup's scheme is provably secure.

HISTORY of DIGITAL SIGNATURES

- In 1976 Diffie and Hellman were first to describe the idea of a digital signature scheme. However, they only conjectured that such schemes may exist.

HISTORY of DIGITAL SIGNATURES

- In 1976 Diffie and Hellman were first to describe the idea of a digital signature scheme. However, they only conjectured that such schemes may exist.
- In 1977 RSA was invented that could be used to produce a primitive (not secure enough) digital signatures.

HISTORY of DIGITAL SIGNATURES

- In 1976 Diffie and Hellman were first to describe the idea of a digital signature scheme. However, they only conjectured that such schemes may exist.
- In 1977 RSA was invented that could be used to produce a primitive (not secure enough) digital signatures.
- The first widely marketed software package to offer digital signature was Lotus Notes 1.0, based on RSA and released in 1989

HISTORY of DIGITAL SIGNATURES

- In 1976 Diffie and Hellman were first to describe the idea of a digital signature scheme. However, they only conjectured that such schemes may exist.
- In 1977 RSA was invented that could be used to produce a primitive (not secure enough) digital signatures.
- The first widely marketed software package to offer digital signature was Lotus Notes 1.0, based on RSA and released in 1989
- ElGamal digital signatures were invented in 1984.

HISTORY of DIGITAL SIGNATURES

- In 1976 Diffie and Hellman were first to describe the idea of a digital signature scheme. However, they only conjectured that such schemes may exist.
- In 1977 RSA was invented that could be used to produce a primitive (not secure enough) digital signatures.
- The first widely marketed software package to offer digital signature was Lotus Notes 1.0, based on RSA and released in 1989
- ElGamal digital signatures were invented in 1984.
- In 1988 Goldwasser, Micali and Rivest were first to rigorously define (perfect) security of digital signature schemes.

APPENDIX

APPENDIX

- Digital signatures are often used to implement electronic signatures - this is a broader term that refers to any electronic data that carries the intend of a signature. Not all electronic signatures use digital signatures.

- Digital signatures are often used to implement electronic signatures - this is a broader term that refers to any electronic data that carries the intend of a signature. Not all electronic signatures use digital signatures.
- In some countries digital signatures have legal significance.

- Digital signatures are often used to implement electronic signatures - this is a broader term that refers to any electronic data that carries the intend of a signature. Not all electronic signatures use digital signatures.
- In some countries digital signatures have legal significance.
- Properly implemented digital signatures are more difficult to forge than the handwritten ones.

- Digital signatures are often used to implement electronic signatures - this is a broader term that refers to any electronic data that carries the intend of a signature. Not all electronic signatures use digital signatures.
- In some countries digital signatures have legal significance.
- Properly implemented digital signatures are more difficult to forge than the handwritten ones.
- Digital signatures can also provide non-repudiation. This means that the signer cannot successfully claim: (a) that he did not signed a message, (b) his private key remain secret.

- Digital signatures are often used to implement electronic signatures - this is a broader term that refers to any electronic data that carries the intend of a signature. Not all electronic signatures use digital signatures.
- In some countries digital signatures have legal significance.
- Properly implemented digital signatures are more difficult to forge than the handwritten ones.
- Digital signatures can also provide non-repudiation. This means that the signer cannot successfully claim: (a) that he did not signed a message, (b) his private key remain secret.

GENERAL OBSERVATIONS - II.

GENERAL OBSERVATIONS - II.

DSA was adopted in US as Federal Information Processing Standard for digital signatures in 1991.

GENERAL OBSERVATIONS - II.

DSA was adopted in US as Federal Information Processing Standard for digital signatures in 1991.

Adaptation was revised in 1996, 2000, 2009 and 2013

GENERAL OBSERVATIONS - II.

DSA was adopted in US as Federal Information Processing Standard for digital signatures in 1991.

Adaptation was revised in 1996, 2000, 2009 and 2013

DSA is covered by US-patent attributed to David W. Krantz (former NSA employee). Claus P. Schnor claims that his US patent covered DSA.

GENERAL OBSERVATIONS - II.

DSA was adopted in US as Federal Information Processing Standard for digital signatures in 1991.

Adaptation was revised in 1996, 2000, 2009 and 2013

DSA is covered by US-patent attributed to David W. Krantz (former NSA employee). Claus P. Schnor claims that his US patent covered DSA.

APPENDIX

GENERAL OBSERVATIONS

- Digital signatures are often used to implement electronic signatures - this is a broader term that refers to any electronic data that carries the intend of a signature. Not all electronic signatures use digital signatures.
- The first broadly marketed software package to offer digital signature was Lotus Notes 1.0, released in 1989, which used RSA algorithm

SECURITY of ElGamal SIGNATURES

Let us analyze several ways an eavesdropper Eve can try to forge ElGamal signature (with x - secret; p, q and $y = q^x \bmod p$ - public):

$$\text{sig}(w, r) = (a, b);$$

where r is random and $a = q^r \bmod p$; $b = (w - xa)r^{-1} \pmod{p-1}$.

SECURITY of ElGamal SIGNATURES

Let us analyze several ways an eavesdropper Eve can try to forge ElGamal signature (with x - secret; p, q and y = q^x mod p - public):

$$\text{sig}(w, r) = (a, b);$$

where r is random and a = q^r mod p; b = $(w - xa)r^{-1} \pmod{p-1}$.

1. First suppose Eve tries to forge signature for a new message w, without knowing x.
   - If Eve first chooses a value a and tries to find the corresponding b, it has to compute the discrete logarithm

     $$lg_a q^w y^{-a},$$

     (because $a^b \equiv q^{r(w-xa)r^{-1}} \equiv q^{w-xa} \equiv q^w y^{-a}$) what is unfeasible.
   - If Eve first chooses b and then tries to find a, she has to solve the equation

     $$y^a a^b \equiv q^{xa} q^{rb} \equiv q^w \pmod{p}.$$

     It is not known whether this equation can be solved for any given b efficiently.

SECURITY of ElGamal SIGNATURES

Let us analyze several ways an eavesdropper Eve can try to forge ElGamal signature (with x - secret; p, q and y = q^x mod p - public):

$$\text{sig}(w, r) = (a, b);$$

where r is random and a = q^r mod p; b = $(w - xa)r^{-1} \pmod{p-1}$.

1. First suppose Eve tries to forge signature for a new message w, without knowing x.
   - If Eve first chooses a value a and tries to find the corresponding b, it has to compute the discrete logarithm

     $$lg_a q^w y^{-a},$$

     (because $a^b \equiv q^{r(w-xa)r^{-1}} \equiv q^{w-xa} \equiv q^w y^{-a}$) what is unfeasible.
   - If Eve first chooses b and then tries to find a, she has to solve the equation

     $$y^a a^b \equiv q^{xa} q^{rb} \equiv q^w \pmod{p}.$$

     It is not known whether this equation can be solved for any given b efficiently.
2. If Eve chooses a and b and tries to determine such w that (a,b) is signature of w, then she has to compute discrete logarithm

   $$lg_q y^a a^b.$$

   Hence, Eve can not sign a "random" message this way.