Diffie-Hellman Key Exchange Protocol and RSA-FDH

Subhabrata Samajder

CREST CRYPTO SUMMER SCHOOL (CCSS), 2025

tcg crest
Inventing Harmonious Future

24th June, 2025

Symmetric-Key Encryption (Recap)

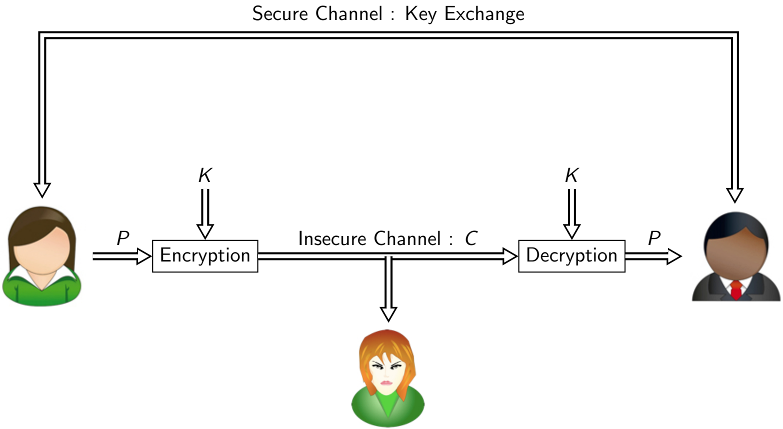


Figure: Symmetric-key Setup

The Discrete-Logarithm Problem (DLog)

Assumptions

- $\mathcal{G}(1^n)$: Denotes a generic, PPT *group generations algorithm*.
    - Outputs a description of a cyclic group $\mathbb{G}$ of order q (with $||q|| \triangleq \lceil \log_2 q \rceil = n$), and a generator $g \in \mathbb{G}$.
- The description of a cyclic group specifies how elements of the group are represented as bit-strings.
- Each group element is represented by a unique bit-string.
- There are *efficient* algorithms for computing the following.
    - The group operation $\circ$ in $\mathbb{G}$.
    - Testing whether a given bit-string represents an element of $\mathbb{G}$.
- Efficient computation of the group operation:
    - Efficient algorithms for exponentiation in $\mathbb{G}$
    - Sampling a uniform element $h \in G$.
        - Choose $x \xleftarrow{\$} \mathbb{Z}_q$.
        - Set $h := g^x$.

Discrete Logarithm

- If $\mathbb{G} = \langle g \rangle$ with $\circ(G) = q$, then $\mathbb{G} = \{g^0, g^1, \ldots, g^{q-1}\}$.
- Equivalently, for every $h \in \mathbb{G}$ there is a unique $x \in \mathbb{Z}_q$, s.t.,

$$g^x = h.$$

- *Discrete logarithm of h with respect to g:* $\log_g h = x$.
- **Note:** If $g^{x'} = h$ for some $x' \in \mathbb{Z}$, then $x' \mod q \equiv \log_g h$.
- **Some Properties:**
    - $\log_g 1 = 0$.
    - $\log_g h^r \equiv (r \cdot \log_g h) \mod q$.
    - $\log_g (h_1 h_2) \equiv (\log_g h_1 + \log_g h_2) \mod q$.

The Discrete-Logarithm Experiment: $\mathrm{DLog}_{\mathcal{A},\mathcal{G}}(n)$

Challenger ($\mathcal{C}$)

Adversary ($\mathcal{A}$)

The Discrete-Logarithm Experiment: $\mathsf{DLog}_{\mathcal{A},\mathcal{G}}(n)$

Challenger ($\mathcal{C}$)

$(\mathbb{G}, q, g) \leftarrow \mathcal{G}(1^n)$

Choose $h \xleftarrow{\$} \mathbb{G}$.

Adversary ($\mathcal{A}$)

The Discrete-Logarithm Experiment: $\mathsf{DLog}_{\mathcal{A},\mathcal{G}}(n)$

Challenger ($\mathcal{C}$)

Adversary ($\mathcal{A}$)

$(\mathbb{G}, q, g) \leftarrow \mathcal{G}(1^n)$

Choose $h \xleftarrow{\$} \mathbb{G}$.

$$\xrightarrow{(\mathbb{G}, q, g, h)}$$

The Discrete-Logarithm Experiment: $\mathsf{DLog}_{\mathcal{A},\mathcal{G}}(n)$

Challenger ($\mathcal{C}$)

Adversary ($\mathcal{A}$)

$(\mathbb{G}, q, g) \leftarrow \mathcal{G}(1^n)$

Choose $h \xleftarrow{\$} \mathbb{G}$.

$\xrightarrow{(\mathbb{G}, q, g, h)}$

$\xleftarrow{x}$

The Discrete-Logarithm Experiment: $\mathsf{DLog}_{\mathcal{A},\mathcal{G}}(n)$

Challenger ($\mathcal{C}$)

Adversary ($\mathcal{A}$)

$(\mathbb{G}, q, g) \leftarrow \mathcal{G}(1^n)$

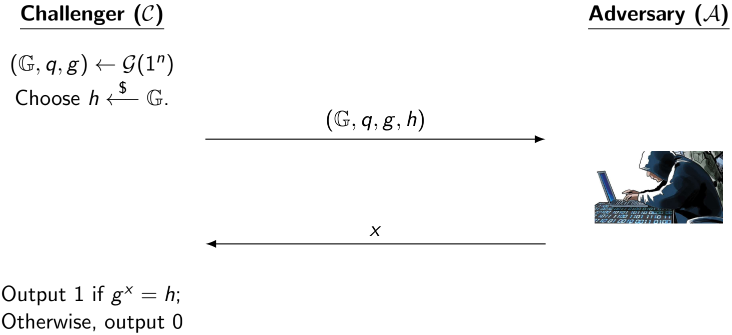
Choose $h \xleftarrow{\$} \mathbb{G}$.

$$\xrightarrow{\qquad (\mathbb{G}, q, g, h) \qquad}$$

$$\xleftarrow{\qquad x \qquad}$$

Output 1 if $g^x = h$;
Otherwise, output 0

The Discrete-Logarithm Experiment: $\mathsf{DLog}_{\mathcal{A},\mathcal{G}}(n)$



Challenger ($\mathcal{C}$)

Adversary ($\mathcal{A}$)

$(\mathbb{G}, q, g) \leftarrow \mathcal{G}(1^n)$

Choose $h \xleftarrow{\$} \mathbb{G}$.

$(\mathbb{G}, q, g, h)$

x

Output 1 if $g^x = h$;
Otherwise, output 0

Definition (Discrete-Logarithm Assumption)

We say that the discrete-logarithm problem is hard relative to $\mathcal{G}$ if for all PPT algorithms $\mathcal{A}$ there exists a negligible function negl, s.t.,

$$\Pr[\mathsf{DLog}_{\mathcal{A},\mathcal{G}}(n) = 1] \leq \mathsf{negl}(n).$$

The Diffie-Hellman Key-exchange Protocol

Using Asymmetry for Key Exchange

- Until 1976, it was generally believed that secure communication could not be achieved without first sharing some secret information using a private channel.
- Whitfield Diffie and Martin Hellman. IEEE-IT (1976), "*New Directions in Cryptography*".
  - Observed that there is often *asymmetry* in the world.
  - Certain actions can be performed easily but cannot be easily reversed.
  - **Example:**
    - Padlocks can be locked without a key, but then cannot be re-opened.
    - It is easy to shatter a glass vase but extremely difficult to put it back together again.
    - Factorization Problem: It is easy to multiply two large primes but difficult to recover those primes from their product.

The Setting

- Alice and Bob: Both runs a probabilistic protocol Π in order to generate a shared, secret key.
  - Π: The set of instructions for Alice and Bob in the protocol.
  - Alice and Bob begin by holding the security parameter 1^n.
  - They then run Π using *independent* random bits.
  - At the end of the protocol, Alice and Bob output keys $k_A, k_B \in \{0,1\}^n$, respectively.
  - **Correctness:** $k_A = k_B = k$ (say).

Definition of Security

- **Intuitive:** A key-exchange protocol is secure if the key output by Alice and Bob is completely *unguessable* by an eavesdropping adversary.

- **Formally:** An adversary eavesdropping on an execution of the protocol should be unable to distinguish the key k generated by Π from a uniform key of length n.

Definition of Security

- **Note:**
  - This is much stronger than simply requiring that the adversary be unable to compute k exactly.
  - But it is necessary since the parties will subsequently use k for some cryptographic application (e.g., as a key for a private-key encryption scheme).

The key-exchange experiment $\mathsf{KE}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n)$

1. Two parties holding 1^n execute protocol Π. This results in the following outputs.
    - trans: Contains all the messages sent by the parties.
    - k: Output of each of the parties.
2. Choose $b \xleftarrow{\$} \{0,1\}$.
    - If $b = 0$ set $\hat{k} := k$.
    - If $b = 1$ then choose $\hat{k} \xleftarrow{\$} \{0,1\}^n$.
3. $\mathcal{A}$ is given trans and $\hat{k}$, and outputs a bit b'.
4. Output 1 if $b' = b$, and 0 otherwise.

In case $\mathsf{KE}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n) = 1$, we say that $\mathcal{A}$ succeeds.

The key-exchange experiment $\mathsf{KE}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n)$

Definition (EAV-secure Key-exchange Protocol)

A key-exchange protocol Π is secure in the presence of an eavesdropper if for all PPT adversaries $\mathcal{A}$ there is a negligible function negl such that

$$\Pr[\mathsf{KE}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n) = 1] \leq \frac{1}{2} + \mathsf{negl}(n).$$

The Diffie-Hellman Key-exchange Protocol

$\mathcal{G}$: Is a PPT algorithm that, on input 1^n, outputs a description of a cyclic group $\mathbb{G}$ of order q (with $||q|| = n$), and a generator $g \in \mathbb{G}$.

The Diffie-Hellman Key-exchange Protocol

Alice

Bob

The Diffie-Hellman Key-exchange Protocol

Alice

Bob

$x \xleftarrow{\$} \mathbb{Z}_q$

The Diffie-Hellman Key-exchange Protocol

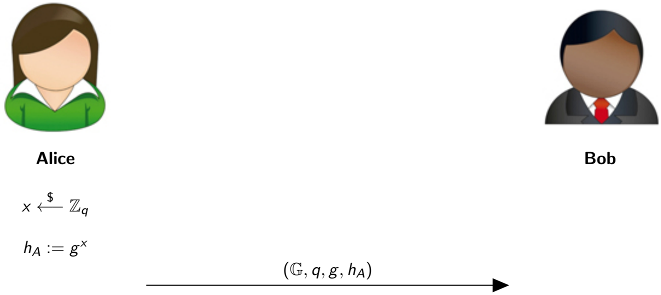
Alice

Bob

$x \xleftarrow{\$} \mathbb{Z}_q$

$h_A := g^x$

The Diffie-Hellman Key-exchange Protocol



Alice

Bob

$x \xleftarrow{\$} \mathbb{Z}_q$

$h_A := g^x$

$$(\mathbb{G}, q, g, h_A)$$

The Diffie-Hellman Key-exchange Protocol

Alice

Bob

$x \xleftarrow{\$} \mathbb{Z}_q$

$h_A := g^x$

$$\xrightarrow{(\mathbb{G}, q, g, h_A)}$$

$y \xleftarrow{\$} \mathbb{Z}_q$

The Diffie-Hellman Key-exchange Protocol

Alice

Bob

$x \xleftarrow{\$} \mathbb{Z}_q$

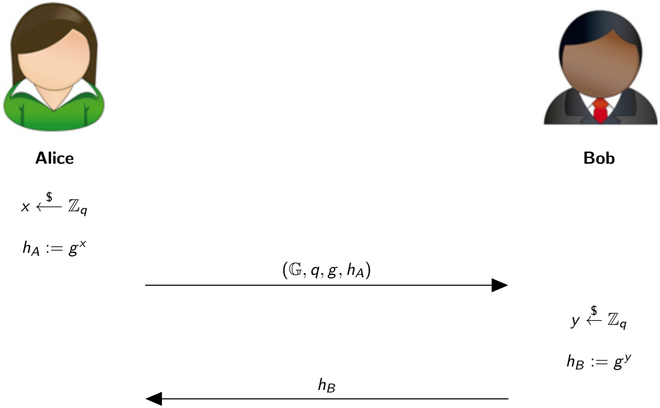
$h_A := g^x$

$$\xrightarrow{\quad (\mathbb{G}, q, g, h_A) \quad}$$

$y \xleftarrow{\$} \mathbb{Z}_q$

$h_B := g^y$

The Diffie-Hellman Key-exchange Protocol



Alice

$x \xleftarrow{\$} \mathbb{Z}_q$

$h_A := g^x$

$(\mathbb{G}, q, g, h_A)$

Bob

$y \xleftarrow{\$} \mathbb{Z}_q$

$h_B := g^y$

h_B

The Diffie-Hellman Key-exchange Protocol

Alice

Bob

Alice: $x \xleftarrow{\$} \mathbb{Z}_q$

Alice: $h_A := g^x$

$$\xrightarrow{(\mathbb{G}, q, g, h_A)}$$

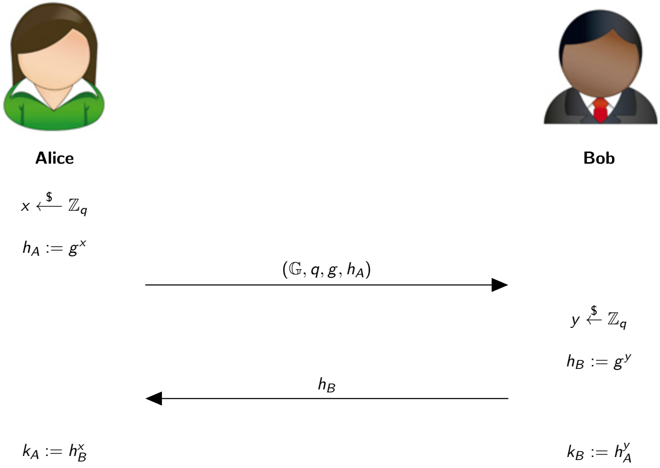
Bob: $y \xleftarrow{\$} \mathbb{Z}_q$

Bob: $h_B := g^y$

$$\xleftarrow{h_B}$$

Bob: $k_B := h_A^y$

The Diffie-Hellman Key-exchange Protocol



Alice

Bob

$x \xleftarrow{\$} \mathbb{Z}_q$

$h_A := g^x$

$(\mathbb{G}, q, g, h_A)$

$y \xleftarrow{\$} \mathbb{Z}_q$

$h_B := g^y$

h_B

$k_A := h_B^x$

$k_B := h_A^y$

Assumptions Needed to Prove Security

- **Minimal security requirement:** Discrete-logarithm problem (DLog) should be hard relative to $\mathcal{G}$.
  - If not, then $\mathcal{A}$ given trans (which, includes h_A) can compute $\log_g h_A = \log_g g^x = x$, the secret value of Alice.
  - Then the shared key $= h_B^x$.
  - Hardness of the DLog is *necessary* for the protocol to be secure.
  - It is however, not sufficient.

Assumptions Needed to Prove Security

- **Note:**
  - There could be other ways of computing the shared key k without explicitly computing x or y.
  - *Computational* Diffie-Hellman (CDH) assumption: Guarantees that the key g^{xy} is hard to compute in its entirety from trans.
  - But CDH does not suffice either.
  - What is required is that the shared key g^{xy} should be *indistinguishable from uniform* for any adversary given g, g^x, and g^y - *decisional* Diffie-Hellman (DDH) assumption.

The Diffie-Hellman Problems

Diffie-Hellman Problems and DLog

- The Diffie-Hellman problems are related, but not known to be equivalent to DLog.

- There are two important variants:
    - Computational Diffie-Hellman (CDH) problem
    - Decisional Diffie-Hellman (DDH) problem

Computational Diffie-Hellman (CDH) Problem

- Fix a cyclic group $\mathbb{G}$ and a generator $g \in \mathbb{G}$.
- Given elements $h_1, h_2 \in \mathbb{G}$, define

$$\mathsf{DH}_g(h_1, h_2) \triangleq g^{(\log_g h_1)\cdot(\log_g h_2)}.$$

- That is, if $h_1 = g^{x_1}$ and $h_2 = g^{x_2}$ then

$$\mathsf{DH}_g(h_1, h_2) = g^{x_1 \cdot x_2} = h_1^{x_2} = h_2^{x_1}.$$

- CDH problem: Compute $\mathsf{DH}_g(h_1, h_2)$ for uniform h_1 and h_2.

Decisional Diffie-Hellman (DDH) Problem

Definition

We say that the DDH problem is hard relative to $\mathcal{G}$ if for all PPT algorithms $\mathcal{A}$ there is a negligible function negl, s.t.,

$$|\Pr[\mathcal{A}(\mathbb{G}, q, g, g^x, g^y, g^z) = 1] - \Pr[\mathcal{A}(\mathbb{G}, q, g, g^x, g^y, g^{xy}) = 1]| \\ \leq \text{negl}(n),$$

where in each case the probabilities are taken over the experiment in which $\mathcal{G}(1^n)$ outputs $(\mathbb{G}, q, g)$, and then uniform $x, y, z \in \mathbb{Z}_q$ are chosen.

Recall that when $z \xleftarrow{\$} \mathbb{Z}_q$, then g^z is uniformly distributed in $\mathbb{G}$.

Security Proof of Diffie-Hellman Key-Exchange Protocol

Theorem 1

Theorem

If the decisional Diffie-Hellman (DDH) *problem is hard relative to* $\mathcal{G}$*, then the Diffie-Hellman key-exchange protocol* Π *is secure in the presence of an eavesdropper .*

Proof of Theorem 1

Let $\mathcal{A}$ be a PPT adversary.

Since $\Pr[b = 0] = \Pr[b = 1] = 1/2$, we have

$$\begin{aligned} & \Pr[\widehat{\mathsf{KE}}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n) = 1] \\ = \quad & \frac{1}{2} \cdot \Pr[\widehat{\mathsf{KE}}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n) = 1 | b = 0] + \frac{1}{2} \cdot \Pr[\widehat{\mathsf{KE}}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n) = 1 | b = 1] \end{aligned}$$

Recall that $\mathcal{A}$ receives $(\underbrace{\mathbb{G}, q, g, h_A, h_B}_{\mathsf{trans}}, \hat{k})$, where $\hat{k}$ is either the actual key computed by the parties (if $b = 0$) or a uniform group element (if $b = 1$).

Proof of Theorem 1

Now, distinguishing between these two cases is exactly equivalent to solving the DDH, i.e.,

$$\begin{aligned}
&\Pr[\widehat{\mathsf{KE}}^{eav}_{\mathcal{A},\Pi}(n) = 1] \\
= \;& \frac{1}{2}\left(\Pr[\widehat{\mathsf{KE}}^{eav}_{\mathcal{A},\Pi}(n) = 1 | b = 0] + \Pr[\widehat{\mathsf{KE}}^{eav}_{\mathcal{A},\Pi}(n) = 1 | b = 1]\right) \\
= \;& \frac{1}{2}(\Pr[\mathcal{A}(\mathbb{G}, g, q, g^x, g^y, g^{xy}) = 0] + \\
& \Pr[\mathcal{A}(\mathbb{G}, g, q, g^x, g^y, g^z) = 1]) \\
= \;& \frac{1}{2}(1 - \Pr[\mathcal{A}(\text{trans}, g^{xy}) = 1]) + \frac{1}{2} \cdot \Pr[\mathcal{A}(\text{trans}, g^z) = 1] \\
= \;& \frac{1}{2} + \frac{1}{2} \cdot (\Pr[\mathcal{A}(\text{trans}, g^z) = 1] - \Pr[\mathcal{A}(\text{trans}, g^{xy}) = 1])
\end{aligned}$$

Proof of Theorem 1

Now, distinguishing between these two cases is exactly equivalent to solving the DDH, i.e.,

$$\begin{aligned}
&\Pr[\widehat{\mathsf{KE}}^{eav}_{\mathcal{A},\Pi}(n) = 1] \\
\leq \quad & \frac{1}{2} + \frac{1}{2} \cdot |\Pr[\mathcal{A}(\mathsf{trans}, g^z) = 1] - \Pr[\mathcal{A}(\mathsf{trans}, g^{xy}) = 1]| \\
& \text{[By triangle inequality]}, \qquad (1)
\end{aligned}$$

where the probabilities are all taken over $(\mathbb{G}, q, g)$ output by $\mathcal{G}(1^n)$, and uniform choice of $x, y, z \in Z_q$.

Proof of Theorem 1

Now DDH is hard relative to $\mathcal{G}$, implies that

$$|\Pr[\mathcal{A}(\mathsf{trans}, g^z) = 1] - \Pr[\mathcal{A}(\mathsf{trans}, g^{xy}) = 1]| \ \leq \ \mathsf{negl}(n).$$

Then from (1), we get

$$\Pr[\widehat{\mathsf{KE}}^{eav}_{\mathcal{A},\Pi}(n) = 1] \leq \frac{1}{2} + \frac{1}{2} \cdot \mathsf{negl}(n),$$

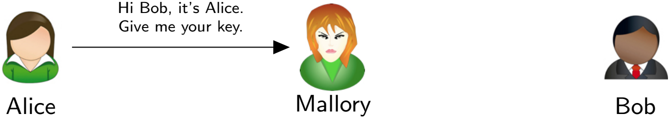
completing the proof.

Why Authentication?

Meet-in-the-Middle Attack (MITM)

Meet-in-the-Middle Attack (MITM)

Meet-in-the-Middle Attack (MITM)



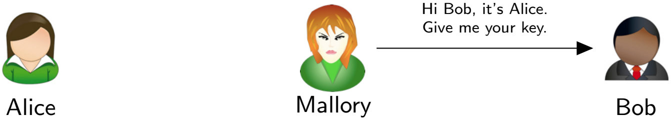
Hi Bob, it's Alice.
Give me your key.

Alice

Mallory

Bob

Meet-in-the-Middle Attack (MITM)



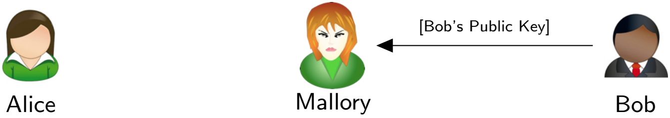
Hi Bob, it's Alice.
Give me your key.

Alice

Mallory

Bob

Meet-in-the-Middle Attack (MITM)



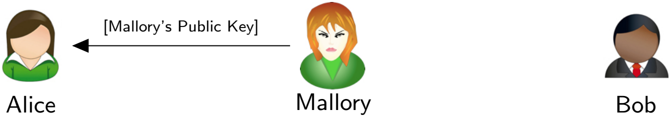
Alice

Mallory

[Bob’s Public Key]

Bob

Meet-in-the-Middle Attack (MITM)



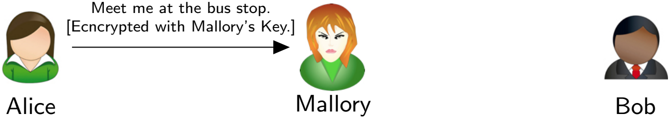
[Mallory's Public Key]

Alice

Mallory

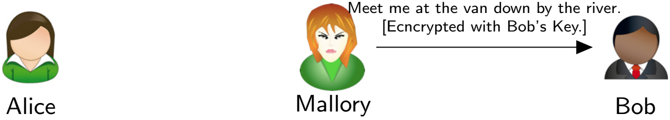
Bob

Meet-in-the-Middle Attack (MITM)



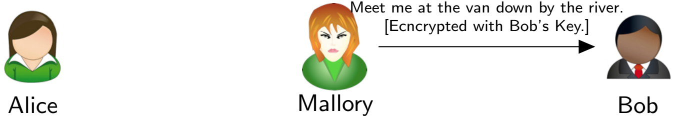
Meet me at the bus stop.
[Ecncrypted with Mallory's Key.]

Meet-in-the-Middle Attack (MITM)



Meet me at the van down by the river.
[Ecncrypted with Bob's Key.]

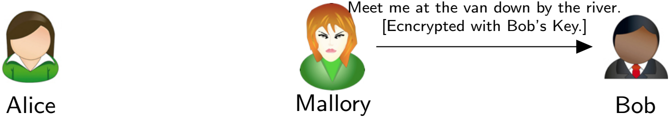
Meet-in-the-Middle Attack (MITM)



Bob:

- Thinks the message is a secure communication from Alice.
- Goes to the van down by the river.
- Gets robbed by Mallory.

Meet-in-the-Middle Attack (MITM)



Alice

Mallory

Meet me at the van down by the river.
[Ecncrypted with Bob's Key.]

Bob

Bob:

- Thinks the message is a secure communication from Alice.
- Goes to the van down by the river.
- Gets robbed by Mallory.

Alice:

- Does not know that Bob was robbed by Mallory.
- Thinks Bob will not come.
- Therefore goes home.

Meet-in-the-Middle Attack (MITM)

- Alice and Bob needs some way to ensure that they are truly using each other's public keys.
- And not the public key of an attacker.
- Otherwise, such attacks are generally possible.

Meet-in-the-Middle Attack (MITM)

- Alice and Bob needs some way to ensure that they are truly using each other's public keys.
- And not the public key of an attacker.
- Otherwise, such attacks are generally possible.
- **Defense and detection:**

Meet-in-the-Middle Attack (MITM)

- Alice and Bob needs some way to ensure that they are truly using each other's public keys.
- And not the public key of an attacker.
- Otherwise, such attacks are generally possible.
- **Defense and detection:**
  - **Authentication:**

Meet-in-the-Middle Attack (MITM)

- Alice and Bob needs some way to ensure that they are truly using each other's public keys.
- And not the public key of an attacker.
- Otherwise, such attacks are generally possible.
- **Defense and detection:**
  - **Authentication:** Provides some degree of certainty that a given message has come from a legitimate source.

Meet-in-the-Middle Attack (MITM)

- Alice and Bob needs some way to ensure that they are truly using each other's public keys.
- And not the public key of an attacker.
- Otherwise, such attacks are generally possible.
- **Defense and detection:**
  - **Authentication:** Provides some degree of certainty that a given message has come from a legitimate source.
  - **Tamper Detection:**

Meet-in-the-Middle Attack (MITM)

- Alice and Bob needs some way to ensure that they are truly using each other's public keys.
- And not the public key of an attacker.
- Otherwise, such attacks are generally possible.
- **Defense and detection:**
  - **Authentication:** Provides some degree of certainty that a given message has come from a legitimate source.
  - **Tamper Detection:** Latency examination can potentially detect the attack in certain situations.

Meet-in-the-Middle Attack (MITM)

- Alice and Bob needs some way to ensure that they are truly using each other's public keys.
- And not the public key of an attacker.
- Otherwise, such attacks are generally possible.
- **Defense and detection:**
    - **Authentication:** Provides some degree of certainty that a given message has come from a legitimate source.
    - **Tamper Detection:** Latency examination can potentially detect the attack in certain situations.
    - **Forensic analysis:**

Meet-in-the-Middle Attack (MITM)

- Alice and Bob needs some way to ensure that they are truly using each other's public keys.
- And not the public key of an attacker.
- Otherwise, such attacks are generally possible.
- **Defense and detection:**
    - **Authentication:** Provides some degree of certainty that a given message has come from a legitimate source.
    - **Tamper Detection:** Latency examination can potentially detect the attack in certain situations.
    - **Forensic analysis:** *Captured network traffic* from what is suspected to be an attack are analyzed.

Meet-in-the-Middle Attack (MITM)

- Alice and Bob needs some way to ensure that they are truly using each other's public keys.
- And not the public key of an attacker.
- Otherwise, such attacks are generally possible.
- **Defense and detection:**
  - **Authentication:** Provides some degree of certainty that a given message has come from a legitimate source.
  - **Tamper Detection:** Latency examination can potentially detect the attack in certain situations.
  - **Forensic analysis:** *Captured network traffic* from what is suspected to be an attack are analyzed.

Public-Key Encryption (Recap)

Alice

Bob

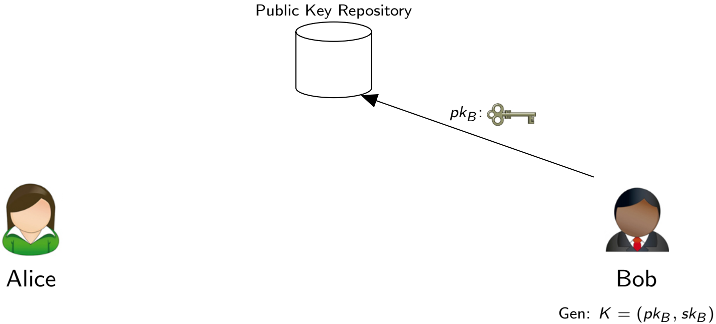
Public-Key Encryption (Recap)

Alice

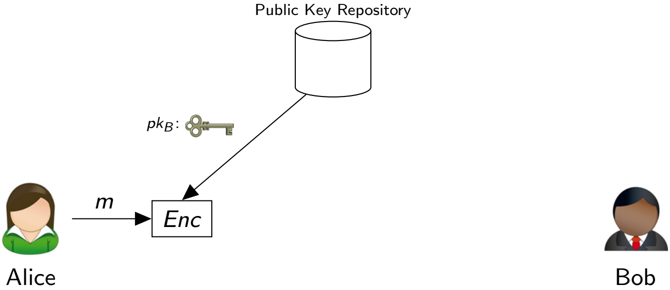
Bob

Gen: $K = (pk_B, sk_B)$

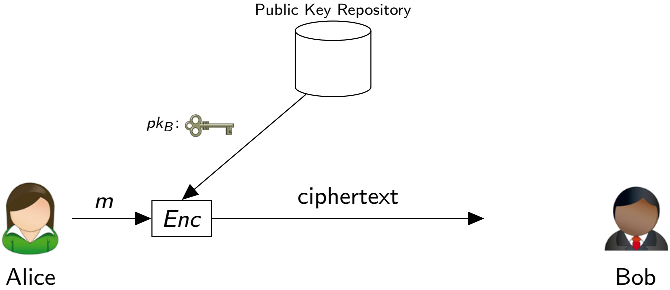
Public-Key Encryption (Recap)



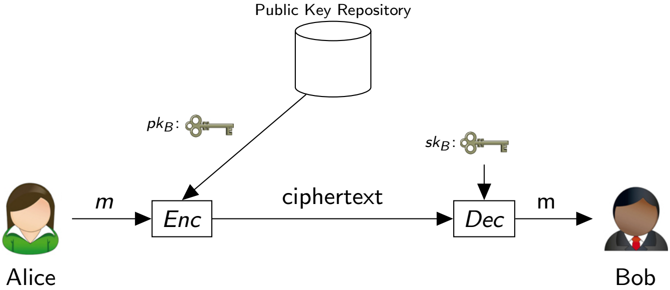
Public-Key Encryption (Recap)



Public-Key Encryption (Recap)



Public-Key Encryption (Recap)



Digital Signatures

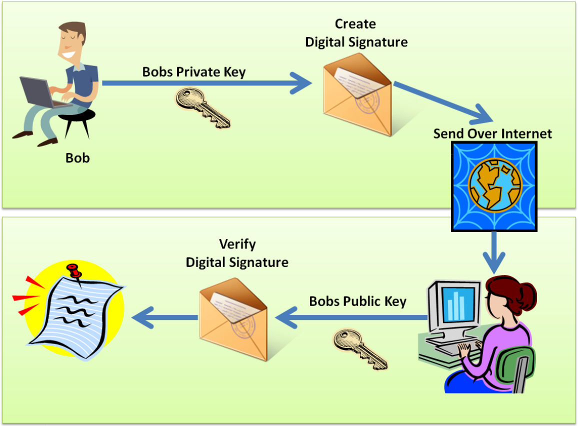
Digital Signatures

- Public-key encryption: Achieves *secrecy* in the PK setting.

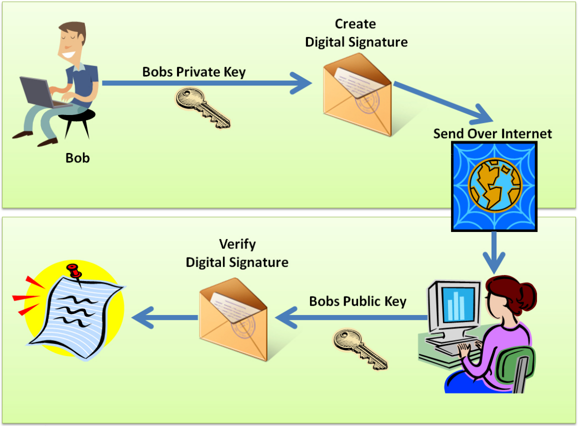
- Digital signature: Provides *Integrity* (or *authenticity*) in the PK setting.

- They are the Public-key analogue of the MACs.

Digital Signatures



Digital Signatures



Note: The *owner of the public key acts as the sender*.

An Example: Software Distribution

Scenario:

- A software company that wants to disseminate software updates in an authenticated manner.

- Mallory: Should not be able to *fool a client* into accepting an update that was not actually released by the company.

An Example: Software Distribution

Digital Signature Solution:

- Company:
    - Generates (pk, sk).
    - Distributes pk in some reliable manner to its clients.
        - **Example:** Bundle pk with the original software purchased by a client.
    - Keeps sk secret.

An Example: Software Distribution

Digital Signature Solution:

- Company:
    - Generates (pk, sk).
    - Distributes pk in some reliable manner to its clients.
        - **Example:** Bundle pk with the original software purchased by a client.
    - Keeps sk secret.
- Release of a software update m:
    - Computes a digital signature σ on m using its private key sk.
    - Sends (m, σ) to every client.

An Example: Software Distribution

Digital Signature Solution:

- Company:
    - Generates (pk, sk).
    - Distributes pk in some reliable manner to its clients.
        - **Example:** Bundle pk with the original software purchased by a client.
    - Keeps sk secret.
- Release of a software update m:
    - Computes a digital signature σ on m using its private key sk.
    - Sends (m, σ) to every client.
- Each Client:
    - Uses pk to verify that σ is a valid signature on m.

An Example: Software Distribution

Digital Signature Solution:

- Mallory: Might try to issue a fraudulent update by sending (m', σ') to a client, where $m' \neq m$ - *forgery*.

An Example: Software Distribution

Digital Signature Solution:

- Mallory: Might try to issue a fraudulent update by sending (m', σ') to a client, where $m' \neq m$ - *forgery*.

- **“Secure”:**
  - If client's attempts to verify the signature σ' on m' fails w.r.t. pk - *invalid signature*.

  - Rejects the signature and therefore the message m'.

Definition

Definition (Digital Signature Scheme)

A **(digital) signature scheme** consists of three PPT algorithms (Gen, Sign, Vrfy) such that:

1. Gen: $(pk, sk) \leftarrow \mathsf{Gen}(1^n)$.
2. Sign: $\sigma \leftarrow \mathsf{Sign}_{sk}(m)$.
3. Vrfy: $b := \mathsf{Vrfy}_{pk}(m, \sigma)$.
   Valid if $b = 1$, else **invalid**.

It is required that except with negligible probability over (pk, sk), it holds that

$$\mathsf{Vrfy}_{pk}(m, \mathsf{Sign}_{sk}(m)) = 1$$

for every (legal) message m.

Digital Signature Model

Signer (S)

Verifier (V)

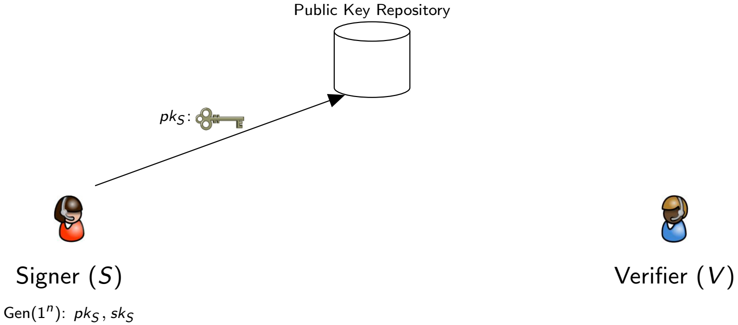
Digital Signature Model

Signer (S)

Gen(1^n): pk_S, sk_S

Verifier (V)

Digital Signature Model



Public Key Repository

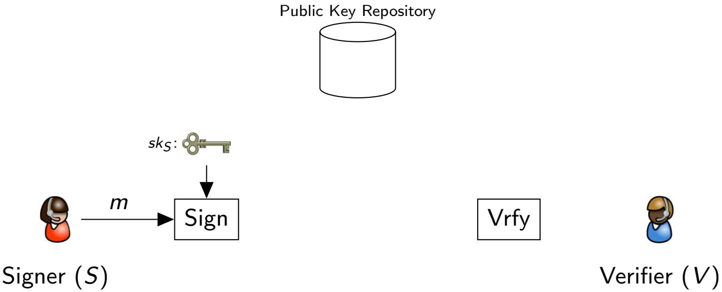
pk_S:

Signer (S)

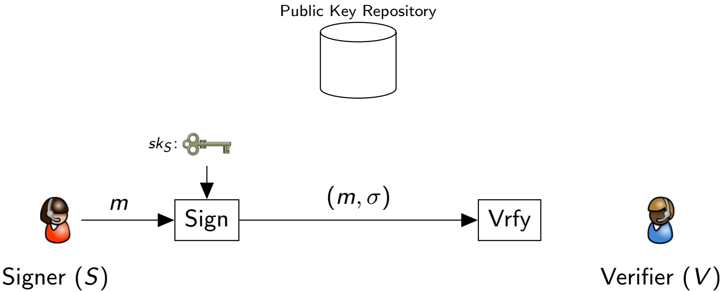
Gen(1^n): pk_S, sk_S

Verifier (V)

Digital Signature Model



Digital Signature Model



Public Key Repository

sk_S:

m

Sign

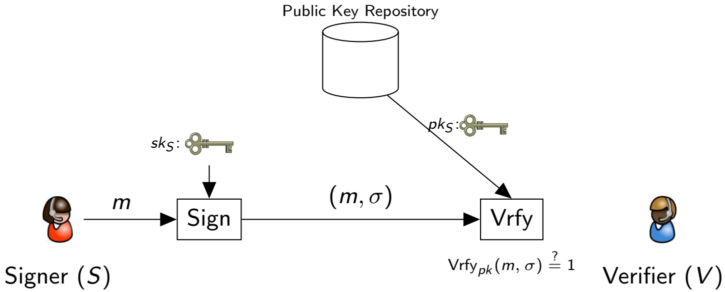
(m, σ)

Vrfy

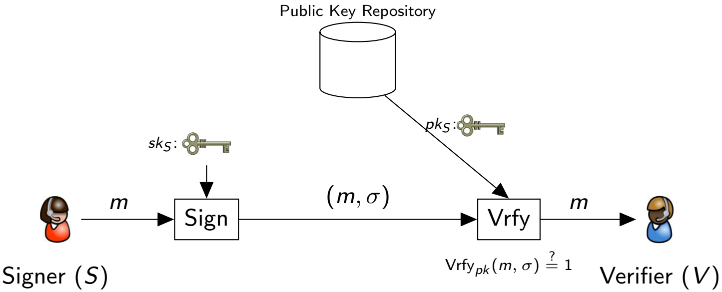
Signer (S)

Verifier (V)

Digital Signature Model



Digital Signature Model



Digital Signature Model

- This establishes that
  - S has sent m, and
  - that m was not modified in transit.

Digital Signature Model

- This establishes that
    - S has sent m, and
    - that m was not modified in transit.

- Unlike MAC, it does not say anything about *when* m was sent.

Digital Signature Model

- This establishes that
    - S has sent m, and
    - that m was not modified in transit.
- Unlike MAC, it does not say anything about *when* m was sent.
- Therefore *replay attacks* are still possible.

Digital Signature Model

- This establishes that
  - S has sent m, and
  - that m was not modified in transit.

- Unlike MAC, it does not say anything about *when* m was sent.

- Therefore *replay attacks* are still possible.

- **Assumption:** The parties are able to obtain a legitimate copy of S's public key.

Digital Signature Model

- This establishes that
  - S has sent m, and
  - that m was not modified in transit.

- Unlike MAC, it does not say anything about *when* m was sent.

- Therefore *replay attacks* are still possible.

- **Assumption:** The parties are able to obtain a legitimate copy of S's public key.
  - Implies that S is able to transmit at least one message (namely, pk itself) in a reliable and authenticated manner.

Digital Signature Model

- This establishes that
  - S has sent m, and
  - that m was not modified in transit.
- Unlike MAC, it does not say anything about *when* m was sent.
- Therefore *replay attacks* are still possible.
- **Assumption:** The parties are able to obtain a legitimate copy of S's public key.
  - Implies that S is able to transmit at least one message (namely, pk itself) in a reliable and authenticated manner.
  - If one then why not all?

Digital Signature Model

- This establishes that
  - S has sent m, and
  - that m was not modified in transit.

- Unlike MAC, it does not say anything about *when* m was sent.

- Therefore *replay attacks* are still possible.

- **Assumption:** The parties are able to obtain a legitimate copy of S's public key.
  - Implies that S is able to transmit at least one message (namely, pk itself) in a reliable and authenticated manner.
  - If one then why not all?
  - In other words, why do we need a signature scheme at all?

Digital Signature Model

Answer:

- Reliable distribution of *pk* is a difficult and expensive task.

Digital Signature Model

Answer:

- Reliable distribution of pk is a difficult and expensive task.

- Signatures ensures that this needs be carried out *only once*.

Digital Signature Model

Answer:

- Reliable distribution of *pk* is a difficult and expensive task.
- Signatures ensures that this needs be carried out *only once*.
- After that an unlimited number of messages can subsequently be sent in a reliable manner.

Digital Signature Model

Answer:

- Reliable distribution of *pk* is a difficult and expensive task.
- Signatures ensures that this needs be carried out *only once*.
- After that an unlimited number of messages can subsequently be sent in a reliable manner.
- Also, signature schemes are used to ensure the reliable distribution of other public keys.

Digital Signature Model

Answer:

- Reliable distribution of *pk* is a difficult and expensive task.
- Signatures ensures that this needs be carried out *only once*.
- After that an unlimited number of messages can subsequently be sent in a reliable manner.
- Also, signature schemes are used to ensure the reliable distribution of other public keys.
- They thus serve as a central tool for setting up a "public-key infrastructure (PKI)" to address the key-distribution problem.

The Signature Experiment Sig-forge$_{\mathcal{A},\Pi}(n)$

For a fixed public key pk generated by a signer S, a **forgery** is a message m along with a valid signature σ, where m was not previously signed by S.

The Signature Experiment Sig-forge$_{\mathcal{A},\Pi}(n)$

For a fixed public key *pk* generated by a signer S, a **forgery** is a message m along with a valid signature σ, where m was not previously signed by S.

Adversary ($\mathcal{A}$)

Verifier (V)

The Signature Experiment Sig-forge$_{\mathcal{A},\Pi}(n)$

For a fixed public key pk generated by a signer S, a **forgery** is a message m along with a valid signature σ, where m was not previously signed by S.

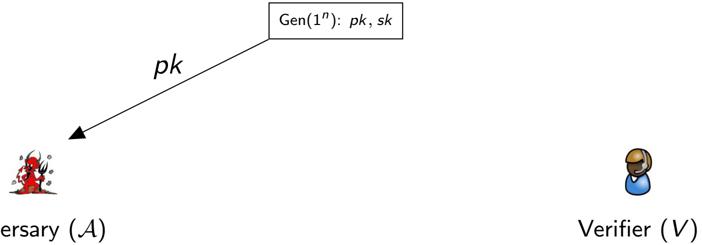
Gen(1^n): pk, sk

Adversary ($\mathcal{A}$)

Verifier (V)

The Signature Experiment Sig-forge$_{\mathcal{A},\Pi}(n)$

For a fixed public key pk generated by a signer S, a **forgery** is a message m along with a valid signature σ, where m was not previously signed by S.



Gen(1^n): pk, sk

pk

Adversary ($\mathcal{A}$)

Verifier (V)

The Signature Experiment Sig-forge$_{\mathcal{A},\Pi}(n)$

For a fixed public key *pk* generated by a signer S, a **forgery** is a message m along with a valid signature σ, where m was not previously signed by S.

Sign_{sk}

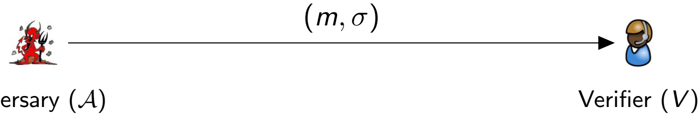
$\mathcal{Q}$

Adversary ($\mathcal{A}$)

Verifier (V)

The Signature Experiment Sig-forge$_{\mathcal{A},\Pi}(n)$

For a fixed public key *pk* generated by a signer S, a **forgery** is a message m along with a valid signature σ, where m was not previously signed by S.



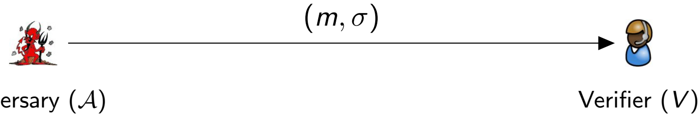
(m, σ)

Adversary ($\mathcal{A}$)

Verifier (V)

The Signature Experiment Sig-forge$_{\mathcal{A},\Pi}(n)$

For a fixed public key *pk* generated by a signer S, a **forgery** is a message m along with a valid signature σ, where m was not previously signed by S.



(m, σ)

Adversary $(\mathcal{A})$

Verifier (V)

$\mathcal{A}$ wins if:

1. $\mathrm{Vrfy}_{\mathrm{pk}}(\mathrm{m}, \sigma) = 1$ and
2. $m \notin \mathcal{Q}$.

The Signature Experiment $\text{Sig-forge}_{\mathcal{A},\Pi}(n)$

Let $\Pi = (\text{Gen}, \text{Sign}, \text{Vrfy})$ be a signature scheme.

1. Run $(pk, sk) \leftarrow \text{Gen}(1^n)$.
2. Adversary $\mathcal{A}$ is given pk and access to an oracle $\text{Sign}_{sk}(\cdot)$. The adversary then outputs (m, σ). Let $\mathcal{Q}$ denote the set of all queries that $\mathcal{A}$ asked its oracle.
3. $\mathcal{A}$ succeeds if and only if
    1. $\text{Vrfy}_{pk}(m, \sigma) = 1$ and
    2. $m \notin \mathcal{Q}$.

   In this case, **output** 1.

The Signature Experiment $\text{Sig-forge}_{\mathcal{A},\Pi}(n)$

Let $\Pi = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ be a signature scheme.

1. Run $(pk, sk) \leftarrow \mathsf{Gen}(1^n)$.
2. Adversary $\mathcal{A}$ is given pk and access to an oracle $\mathsf{Sign}_{sk}(\cdot)$. The adversary then outputs (m, σ). Let $\mathcal{Q}$ denote the set of all queries that $\mathcal{A}$ asked its oracle.
3. $\mathcal{A}$ succeeds if and only if
    1. $\mathsf{Vrfy}_{pk}(m, \sigma) = 1$ and
    2. $m \notin \mathcal{Q}$.

   In this case, **output** 1.

Definition

A signature scheme $\Pi = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ is **existentially unforgeable under an adaptive chosen-message attack**, or just **secure**, if for all PPT adversaries $\mathcal{A}$, there is a negligible function negl, s.t.,

$$\Pr[\text{Sig-forge}_{\mathcal{A},\Pi}(n) = 1] \leq \mathsf{negl}(n).$$

Plain RSA Signature

Plain RSA Signature

- Gen: $(N, e, d) \leftarrow \mathsf{GenRSA}(1^n)$, where $N = pq$ and $ed \equiv 1 \mod \phi(N)$.
  - *pk*: (N, e)
  - *sk*: (N, p, q, d).

- Sign: On input $sk = (N, d)$ and $m \in \mathbb{Z}_N^*$, compute

$$\sigma := m^d \mod N.$$

- Vrfy: On input $pk = (N, e)$, $m \in \mathbb{Z}_N^*$, and a $\sigma \in \mathbb{Z}_N^*$, output 1 if and only if

$$m \stackrel{?}{=} \sigma^e \mod N.$$

Plain RSA Signature

- Gen: $(N, e, d) \leftarrow \mathsf{GenRSA}(1^n)$, where $N = pq$ and $ed \equiv 1 \mod \phi(N)$.
    - *pk*: (N, e)
    - *sk*: (N, p, q, d).

- Sign: On input $sk = (N, d)$ and $m \in \mathbb{Z}_N^*$, compute

$$\sigma := m^d \mod N.$$

- Vrfy: On input $pk = (N, e)$, $m \in \mathbb{Z}_N^*$, and a $\sigma \in \mathbb{Z}_N^*$, output 1 if and only if

$$m \overset{?}{=} \sigma^e \mod N.$$

Correctness: $\sigma^e = (m^d)^e = m^{ed \mod \phi(N)} = m^1 = m \mod N$.

Secure?

Secure?

- Consider an adversary knowing only the public key (N, e).
- Then computing a valid signature on a message m seems to require solving the RSA problem (since the signature is exactly the e^{th} root of m).

Secure?

- Consider an adversary knowing only the public key (N, e).
- Then computing a valid signature on a message m seems to require solving the RSA problem (since the signature is exactly the e^{th} root of m).
- Unfortunately, this reasoning is incorrect.

Secure?

- Consider an adversary knowing only the public key (N, e).
- Then computing a valid signature on a message m seems to require solving the RSA problem (since the signature is exactly the e^{th} root of m).
- Unfortunately, this reasoning is incorrect.
    - The RSA assumption only implies hardness of computing a signature (i.e., computing an e^{th} root) of a uniform message m.

Secure?

- Consider an adversary knowing only the public key (N, e).
- Then computing a valid signature on a message m seems to require solving the RSA problem (since the signature is exactly the e^{th} root of m).
- Unfortunately, this reasoning is incorrect.
  - The RSA assumption only implies hardness of computing a signature (i.e., computing an e^{th} root) of a uniform message m.
  - Says nothing about hardness of computing a signature on a non-uniform m or on some message m of the attacker's choice.

Secure?

- Consider an adversary knowing only the public key (N, e).
- Then computing a valid signature on a message m seems to require solving the RSA problem (since the signature is exactly the e^{th} root of m).
- Unfortunately, this reasoning is incorrect.
  - The RSA assumption only implies hardness of computing a signature (i.e., computing an e^{th} root) of a uniform message m.
  - Says nothing about hardness of computing a signature on a non-uniform m or on some message m of the attacker's choice.
  - The RSA assumption says nothing about what an attacker might be able to do once it learns signatures on other messages.

Attacks

A no-message attack:

- Given a $pk = (N, e)$, choose $\sigma \xleftarrow{\$} \mathbb{Z}_N^*$.
- Compute $m := \sigma^e \mod N$.
- Then output the forgery (m, σ).

Attacks

Forging a signature on an arbitrary message:

- Say the adversary wants to forge a signature on the message $m \in \mathbb{Z}_N^*$ with respect to the public key $pk = (N, e)$.
- $\mathcal{A}$:
  - Chooses arbitrary $m_1, m_2 \in \mathbb{Z}_N^*$ distinct from m such that
    $$m = m_1 \cdot m_2 \mod N.$$
  - Obtains signatures σ_1, σ_2 on m_1, m_2, respectively.
  - Outputs
    $$\sigma := \sigma_1 \cdot \sigma_2 \mod N$$
    as a valid signature on m.

Attacks

Forging a signature on an arbitrary message:

- Say the adversary wants to forge a signature on the message $m \in \mathbb{Z}_N^*$ with respect to the public key $pk = (N, e)$.
- $\mathcal{A}$:
  - Chooses arbitrary $m_1, m_2 \in \mathbb{Z}_N^*$ distinct from m such that

    $$m = m_1 \cdot m_2 \mod N.$$

  - Obtains signatures σ_1, σ_2 on m_1, m_2, respectively.
  - Outputs

    $$\sigma := \sigma_1 \cdot \sigma_2 \mod N$$

    as a valid signature on m.
- Can be extended to n arbitrary messages.

RSA-FDH

How to prevent these trivial attacks?

- **Idea:** Apply some transformation to messages before signing them.

- That is, the signer now specifies as part of its public key a (deterministic) function H with certain cryptographic properties mapping messages to $\mathbb{Z}_N^*$.

- Sign: $\sigma := H(m)^d \mod N$.

- Vrfy: $\sigma^e \stackrel{?}{=} H(m) \mod N$.

The RSA-FDH signature scheme

- **Gen:** $(N, e, d) \leftarrow \mathsf{GenRSA}(1^n)$
  - *pk*: (N, e)
  - *sk*: (N, d)

  As part of key generation, a function $H : \{0,1\}^* \to \mathbb{Z}_N^*$ is specified, but we leave this implicit.

- **Sign:** On input a $sk = (N, d)$ and a $m \in \{0,1\}^*$, compute

$$\sigma := H(m)^d \mod N.$$

- **Vrfy:** On input a $pk = (N, e)$, a message m, and a signature σ, output 1 if and only if

$$\sigma^e \stackrel{?}{=} H(m) \mod N.$$

Properties H Require

- **Prevent the no-message attack:**
    - It should be infeasible for an attacker to start with σ,
        - compute $\hat{m} := \sigma^e \mod N$, and
        - then find a message m such that $H(m) = \hat{m}$.
    - Thus, H should be hard to invert in some sense.

- **To prevent the second attack:**
    - H must not admit "multiplicative relations".
    - It should be hard to find three messages m, m_1, m_2 with

$$H(m) = H(m_1) \cdot H(m_2) \mod N.$$

- **It must be hard to find collisions in H:**
    - If $H(m_1) = H(m_2)$, then m_1 and m_2 have the same signature.
    - That is forgery becomes *trivial*.

Choice of H

- There is no known way to choose H so that the scheme can be proven to be secure.

- **Theorem:** The signature scheme is security under *random oracle model*, i.e., if H is modeled as a *random oracle* that maps its inputs uniformly onto $\mathbb{Z}_N^*$.

- The scheme in this case is called the RSA full-domain hash (RSA-FDH) signature scheme.

- **Note:** A random function of this sort satisfies the requirements discussed previously.
  - A random function (with large range) is hard to invert.
  - Does not have any easy-to-find multiplicative relations.
  - Is collision resistant.

Security Against No-message Attack

Note:

- The adversary $\mathcal{A}$ cannot request any signatures.
- The adversary is limited to making queries to the random oracle.
- Wlog, we assume that $\mathcal{A}$ always makes exactly q (distinct) queries to H.
- If the adversary outputs a forgery (m, σ) then it had previously queried m to H.

Security Against No-message Attack

Assumption:

- $\mathcal{A}$ is an efficient adversary that carries out a no-message attack.
- $\mathcal{A}$ makes exactly q queries to H.

Construct an efficient algorithm $\mathcal{A}'$ for solving the RSA problem relative to GenRSA.

Given input (N, e, y), algorithm $\mathcal{A}'$

- runs $\mathcal{A}$ on the public key $pk = (N, e)$.
- Let $m_1, \ldots, m_q$ denote the q (distinct) queries that $\mathcal{A}$ makes to H.
- $\mathcal{A}'$ answers these random-oracle queries of $\mathcal{A}$ with uniform elements of $\mathbb{Z}_N^*$ except for one query
  - say, the i^{th} query, chosen uniformly from the oracle queries of $\mathcal{A}$
- This i^{th} query is answered with y itself.

Security Against No-message Attack

From the point of view of $\mathcal{A}$, all its random-oracle queries are answered with uniform elements of $\mathbb{Z}_N^*$.

Recall that y is uniform as well, although it is not chosen by $\mathcal{A}'$, and so $\mathcal{A}$ has no information about i.

Moreover, the view of $\mathcal{A}$ when run as a subroutine by $\mathcal{A}'$ is identically distributed to the view of $\mathcal{A}$ when attacking the original signature scheme.

Security Against No-message Attack

If $\mathcal{A}$ outputs a forgery (m, σ) then, because $m \in \{m_1, \ldots, m_q\}$, with probability $1/q$ we will have $m = m_i$.

In that case,

$$\sigma^e = H(m) = H(m_i) = y \mod N$$

and $\mathcal{A}'$ can output σ as the solution to its given RSA instance (N, e, y).

Conclusion:

- If $\mathcal{A}$ outputs a forgery with probability ϵ, then $\mathcal{A}'$ solves the RSA problem with probability ϵ/q.
- Since q is polynomial, we conclude that ϵ must be negligible if the RSA problem is hard relative to GenRSA.

Security: Main Result

Theorem

If the RSA problem is hard relative to GenRSA *and H is modeled as a random oracle, then RSA-FDH is secure.*

The Hash-and-Sign Paradigm

Motivation

- Public-key signature schemes are less efficient than MACs.

- But it is possible to obtain the functionality of digital signatures at the asymptotic cost of a private-key operation, at least for sufficiently long messages.

- This can be done using the *hash-and-sign* paradigm.

Main Idea

- Suppose we have a signature scheme for messages of length ℓ.

- But we wish to sign a (longer) message $m \in \{0,1\}^*$.

- Rather than sign m itself, one can instead use a hash function $H : \{0,1\}^* \to \{0,1\}^\ell$ and then sign the resulting digest.

- This is exactly analogous to the *hash-and-MAC* approach.

Construction 1: The Hash-and-Sign Paradigm

Let $\Pi = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ be a signature scheme for messages of length $\ell(n)$, and let $\Pi_H = (\mathsf{Gen}_H, H)$ be a hash function with output length $\ell(n)$. Construct a signature scheme $\Pi' = (\mathsf{Gen}'$, Sign', $\mathsf{Vrfy}')$ as follows:

- Gen′: On input 1^n, run $\mathsf{Gen}(1^n)$ to obtain (pk, sk) and run $\mathsf{Gen}_H(1^n)$ to obtain k. The public key is (pk, k) and the private key is (sk, k).
- Sign′: On input a private key (sk, k) and a message $m \in \{0,1\}^*$, output
$$\sigma \leftarrow \mathsf{Sign}_{sk}(H_k(m)).$$
- Vrfy′: On input a public key (pk, k), a message $m \in \{0,1\}^*$, and a signature σ, output 1 if and only if
$$\mathsf{Vrfy}_{pk}(H_k(m), \sigma) \overset{?}{=} 1.$$

Theorem

Theorem

If Π *is a secure signature scheme for messages of length* $\ell(n)$ *and* Π_H *is collision resistant, then Construction 1 is a secure signature scheme (for arbitrary-length messages).*

Books Consulted

1. *Introduction to Modern Cryptography* by Jonathan Katz and Yehuda Lindell, 2nd Edition, Chapman & Hall/CRC.

Thank You for your kind attention!

Questions!!