Public-key cryptography

Daniel Bosk

School of Computer Science and Communication,
KTH Royal Institute of Technology, Stockholm

Department of Information and Communication Systems,
Mid Sweden University, Sundsvall

6th April 2020

1 Public-key cryptography
- Key-exchange schemes
- Encryption and decryption
- Digital signatures
- Homomorphic properties

Idea

- It's difficult to have to exchange keys in advance.
- What if we could securely exchange keys at a distance?
- If we could do it just before we use them?

Idea

- It's difficult to have to exchange keys in advance.
- What if we could securely exchange keys at a distance?
- If we could do it just before we use them?

Solution (Requirements)

- *We need a problem that is easy for Alice and Bob.*
- *It should be hard for Eve.*

Definition (Discrete Logarithm Problem, DLP)

- Let $\mathbb{Z}_p^*$ be the multiplicative group of residues modulo $p \in \mathbb{N}$, where p is a prime.

$$\text{Given } g, g^x \in \mathbb{Z}_p^*$$
$$\text{Find } x.$$

- I.e. compute $\log_{g \in \mathbb{Z}_p}(g^x)$.

Definition (Discrete Logarithm Problem, DLP)

- Let $\mathbb{Z}_p^*$ be the multiplicative group of residues modulo $p \in \mathbb{N}$, where p is a prime.

$$\begin{aligned} \text{Given} \quad & g, g^x \in \mathbb{Z}_p^* \\ \text{Find} \quad & x. \end{aligned}$$

- I.e. compute $\log_{g \in \mathbb{Z}_p}(g^x)$.

Definition (Diffie-Hellman Problem, DHP[1])

Given $g, g^x, g^y \in \mathbb{Z}_p^*$

Find g^{xy}

Definition (Decisional Diffie-Hellman Problem, DDH)

Given $g, g^x, g^y, g^z \in \mathbb{Z}_p^*$

Decide $z \stackrel{?}{=} xy$

[1] **DiffieHellman**.

Definition (Diffie-Hellman Problem, DHP[1])

Given $g, g^x, g^y \in \mathbb{Z}_p^*$

Find g^{xy}

Definition (Decisional Diffie-Hellman Problem, DDH)

Given $g, g^x, g^y, g^z \in \mathbb{Z}_p^*$

Decide $z \stackrel{?}{=} xy$

[1] **DiffieHellman**.

- If we can solve DLP, then we can solve DHP and DDH too.
- Maybe DHP and DDH can be solved without DLP.
- We don't know yet.
- We usually assume DLP, DHP and DDH are hard.

- If we can solve DLP, then we can solve DHP and DDH too.
- Maybe DHP and DDH can be solved without DLP.
- We don't know yet.
- We usually assume DLP, DHP and DDH are hard.

- If we can solve DLP, then we can solve DHP and DDH too.
- Maybe DHP and DDH can be solved without DLP.
- We don't know yet.
- We usually assume DLP, DHP and DDH are hard.

Exercise

- **DiffieHellman**[2] used DHP to create a key-exchange protocol.
- Take some time to figure out how we can use these problems to achieve what we want.

Reminder

- Alice and Bob want to exchange a secret key.
- Then they can use the key to encrypt their communications.

[2] **DiffieHellman**.

Exercise

- **DiffieHellman**[2] used DHP to create a key-exchange protocol.
- Take some time to figure out how we can use these problems to achieve what we want.

Reminder

- Alice and Bob want to exchange a secret key.
- Then they can use the key to encrypt their communications.

[2] **DiffieHellman**.

Definition (Diffie-Hellman key-exchange)

- Let $g \in \mathbb{Z}_p^*$ (publicly known, e.g. RFC, standard . . .).
- Alice generates random $0 < x < |\mathbb{Z}_p^*|$.
- She sends g^x to Bob.
- Bob generates random $0 < y < |\mathbb{Z}_p^*|$.
- He sends g^y to Alice.
- Alice has x and g, g^y.
- Bob has g, g^x and y.
- They both compute $g^{xy} = (g^y)^x = (g^x)^y$.
- Eve has g, g^x, g^y.
- By DHP she cannot compute g^{xy}.

Definition (Diffie-Hellman key-exchange)

- Let $g \in \mathbb{Z}_p^*$ (publicly known, e.g. RFC, standard ...).
- Alice generates random $0 < x < |\mathbb{Z}_p^*|$.
- She sends g^x to Bob.
- Bob generates random $0 < y < |\mathbb{Z}_p^*|$.
- He sends g^y to Alice.
- Alice has x and g, g^y.
- Bob has g, g^x and y.
- They both compute $g^{xy} = (g^y)^x = (g^x)^y$.
- Eve has g, g^x, g^y.
- By DHP she cannot compute g^{xy}.

Definition (Diffie-Hellman key-exchange)

- Let $g \in \mathbb{Z}_p^*$ (publicly known, e.g. RFC, standard ...).
- Alice generates random $0 < x < |\mathbb{Z}_p^*|$.
- She sends g^x to Bob.
- Bob generates random $0 < y < |\mathbb{Z}_p^*|$.
- He sends g^y to Alice.
- Alice has x and g, g^y.
- Bob has g, g^x and y.
- They both compute $g^{xy} = (g^y)^x = (g^x)^y$.
- Eve has g, g^x, g^y.
- By DHP she cannot compute g^{xy}.

Definition (Diffie-Hellman key-exchange)

- Let $g \in \mathbb{Z}_p^*$ (publicly known, e.g. RFC, standard ...).
- Alice generates random $0 < x < |\mathbb{Z}_p^*|$.
- She sends g^x to Bob.
- Bob generates random $0 < y < |\mathbb{Z}_p^*|$.
- He sends g^y to Alice.
- Alice has x and g, g^y.
- Bob has g, g^x and y.
- They both compute $g^{xy} = (g^y)^x = (g^x)^y$.
- Eve has g, g^x, g^y.
- By DHP she cannot compute g^{xy}.

Definition (Diffie-Hellman key-exchange)

- Let $g \in \mathbb{Z}_p^*$ (publicly known, e.g. RFC, standard ...).
- Alice generates random $0 < x < |\mathbb{Z}_p^*|$.
- She sends g^x to Bob.
- Bob generates random $0 < y < |\mathbb{Z}_p^*|$.
- He sends g^y to Alice.
- Alice has x and g, g^y.
- Bob has g, g^x and y.
- They both compute $g^{xy} = (g^y)^x = (g^x)^y$.
- Eve has g, g^x, g^y.
- By DHP she cannot compute g^{xy}.

Note

- This is not secure as it is.
- g^x, g^y are *not authenticated*!
- Alice can tell the difference between Bob and Eve!

Idea

- Fine, we can use g^{xy} as a key in a cipher.
  - $\mathrm{Enc}(g^{xy})m$, where Enc is a symmetric cipher.
- But shouldn't we be able to include a message directly?

Definition (ElGamal Encryption Scheme[3])

Set-up:

- Let $g \in \mathbb{Z}_p^*$, randomly choose $0 < x < |\mathbb{Z}_p^*|$.
- Alice publishes $\mathbb{Z}_p^*, g, g^x$ to everyone.

Encryption:

- Bob chooses random $0 < y < |\mathbb{Z}_p^*|$ and computes g^y.
- Bob's message $m \in \mathbb{Z}_p^*$.
- He sends $(g^y, m(g^x)^y)$ to Alice.

Decryption:

- Alice computes $(g^y)^{-x}$ and $m(g^x)^y(g^y)^{-x} = m$.

[3] **ElGamal**.

Idea

- Sure, if Bob sends a message to Alice, he's sure she's the only one who can decrypt it.
- Can't we turn this around?
  - Can't Alice use the same system to ensure Bob knows the message came from Alice?

Exercise

- Look at the ElGamal encryption scheme for a bit.
- Try to find a way to 'run it backwards'.

Idea

- Sure, if Bob sends a message to Alice, he's sure she's the only one who can decrypt it.
- Can't we turn this around?
  - Can't Alice use the same system to ensure Bob knows the message came from Alice?

Exercise

- Look at the ElGamal encryption scheme for a bit.
- Try to find a way to 'run it backwards'.

Idea

- Sure, if Bob sends a message to Alice, he's sure she's the only one who can decrypt it.
- Can't we turn this around?
  - Can't Alice use the same system to ensure Bob knows the message came from Alice?

Exercise

- Look at the ElGamal encryption scheme for a bit.
- Try to find a way to 'run it backwards'.

Definition (ElGamal Signature Scheme[4])

Set-up:

- Let $g \in \mathbb{Z}_p^*$ and $\boxed{h \text{ be a one-way function}}$.
- Alice publishes $\mathbb{Z}_p^*, g, g^x$ to everyone.

Signing $m \in \mathbb{Z}_p^*$:

- $\boxed{\text{Alice}}$ chooses random $0 < y < |\mathbb{Z}_p^*|$ and computes $r = g^y \in \mathbb{Z}_p^*$.
- She computes $\boxed{s = (h(m) - xr)y^{-1} \pmod{|\mathbb{Z}_p^*|}.}$
- She sends (r, s) to Bob.

Verification:

- $\boxed{\text{Bob checks if } g^{h(m)} \stackrel{?}{=}_{\mathbb{Z}_p^*} (g^x)^r r^s} =_{\mathbb{Z}_p^*} (g^x)^{g^y} (g^y)^{(h(m) - xg^y)y^{-1}} =_{\mathbb{Z}_p^*} g^{xg^y + h(m) - xg^y}$

[4] **ElGamal**.

Note

- It works without the hash.
- But then we can multiply two messages and still get a valid signature.

Definition (Homomorphism)

A *homomorphism* is a map (function) that preserves structure between two algebraic structures.

Example

- Let $G_1 = (\mathbb{R}, \cdot)$ and $G_2 = (\mathbb{R}, +)$ be groups.
- $g_1, g_1' \in G_1$ and $g_2, g_2' \in G_2$.
- Consider $\log\colon G_1 \to G_2$.
- $\log(g_1 \cdot g_1') = g_2 + g_2'$.

Definition (Homomorphism)

A *homomorphism* is a map (function) that preserves structure between two algebraic structures.

Example

- Let $G_1 = (\mathbb{R}, \cdot)$ and $G_2 = (\mathbb{R}, +)$ be groups.
- $g_1, g_1' \in G_1$ and $g_2, g_2' \in G_2$.
- Consider $\log \colon G_1 \to G_2$.
- $\log(g_1 \cdot g_1') = g_2 + g_2'$.

Definition (Homomorphism)

A *homomorphism* is a map (function) that preserves structure between two algebraic structures.

Example

- Let $G_1 = (\mathbb{R}, \cdot)$ and $G_2 = (\mathbb{R}, +)$ be groups.
- $g_1, g_1' \in G_1$ and $g_2, g_2' \in G_2$.
- Consider $\log \colon G_1 \to G_2$.
- $\log(g_1 \cdot g_1') = g_2 + g_2'$.

Definition (Homomorphism)

A *homomorphism* is a map (function) that preserves structure between two algebraic structures.

Example

- Let $G_1 = (\mathbb{R}, \cdot)$ and $G_2 = (\mathbb{R}, +)$ be groups.
- $g_1, g_1' \in G_1$ and $g_2, g_2' \in G_2$.
- Consider $\log\colon G_1 \to G_2$.
- $\log(g_1 \cdot g_1') = g_2 + g_2'$.

Exercise

The encryption (decryption) function of the ElGamal cryptosystem is a homomorphism, what structure does it preserve?

Example (ElGamal's homomorphism)

- Messages m, m', ciphertexts $(g^y, m \cdot g^{xy}), (g^{y'}, m' \cdot g^{xy'})$.
- Remember: private key x, hence the same.
- Create ciphertext

$$
\begin{aligned}
(g^y g^{y'}, m \cdot g^{xy} \cdot m' \cdot g^{xy'}) &= (g^{y+y'}, m \cdot m' \cdot g^{xy+xy'}) \\
&= (g^{y+y'}, m \cdot m' \cdot g^{x(y+y')}).
\end{aligned}
$$

- Decryption: take $g^{y+y'}$, compute $(g^{y+y'})^x = g^{x(y+y')}$.
- Decryption thus yields $m \cdot m'$.

Example (ElGamal's homomorphism)

- Messages m, m', ciphertexts $(g^y, m \cdot g^{xy}), (g^{y'}, m' \cdot g^{xy'})$.
- Remember: private key x, hence the same.
- Create ciphertext

$$\begin{aligned}(g^y g^{y'}, m \cdot g^{xy} \cdot m' \cdot g^{xy'}) &= (g^{y+y'}, m \cdot m' \cdot g^{xy+xy'}) \\ &= (g^{y+y'}, m \cdot m' \cdot g^{x(y+y')}).\end{aligned}$$

- Decryption: take $g^{y+y'}$, compute $(g^{y+y'})^x = g^{x(y+y')}$.
- Decryption thus yields $m \cdot m'$.

Example (ElGamal's homomorphism)

- Messages m, m', ciphertexts $(g^y, m \cdot g^{xy}), (g^{y'}, m' \cdot g^{xy'})$.
- Remember: private key x, hence the same.
- Create ciphertext

$$\begin{aligned}(g^y g^{y'}, m \cdot g^{xy} \cdot m' \cdot g^{xy'}) &= (g^{y+y'}, m \cdot m' \cdot g^{xy+xy'}) \\ &= (g^{y+y'}, m \cdot m' \cdot g^{x(y+y')}).\end{aligned}$$

- Decryption: take $g^{y+y'}$, compute $(g^{y+y'})^x = g^{x(y+y')}$.
- Decryption thus yields $m \cdot m'$.

Note

- We use a hash function in the signature scheme to counter the homomorphic property.
- $h(m) \cdot h(m') \neq h(m \cdot m')$.
- Without the hash function we could create a valid signature for a new message *without knowing the signature key!*

Note

- We use a hash function in the signature scheme to counter the homomorphic property.
- $h(m) \cdot h(m') \neq h(m \cdot m')$.
- Without the hash function we could create a valid signature for a new message *without knowing the signature key!*

Note

- There are many schemes with different homomorphic properties.
- There is even *fully homomorphic encryption* [**GentryFullyHomomorphicEncryption**].

Homomorphic properties