Cryptographic methods in privacy

Jan Willemsen

March 2020
Groups: intuition
Groups: intuition

Picturès by Alisa Pankova

March 2020
Groups

Definition

Algebraic structure with one binary operation \((G, \bigotimes)\) is called a group if \(\bigotimes\) is associative, has a neutral element and every element has an inverse (i.e. one can divide/subtract).

If \(\bigotimes\) is commutative, the group is called Abelian.

- Examples: regular \(n\)-gon w.r.t. composition of rotations, \((\mathbb{Z}, +)\), \((\mathbb{Q}, +)\), \((\mathbb{R}, +)\).
Groups: more examples

- **Discussion:** Is \((\mathbb{Z}, \cdot)\) a group? What about \((\mathbb{Q}, \cdot)\) and \((\mathbb{R}, \cdot)\)?
Groups: more examples

- Discussion: Is \((\mathbb{Z}, \cdot)\) a group? What about \((\mathbb{Q}, \cdot)\) and \((\mathbb{R}, \cdot)\)?
- Discussion: Can you turn any of them into a group easily?
Groups: more examples

- **Discussion:** Is \((\mathbb{Z}, \cdot)\) a group? What about \((\mathbb{Q}, \cdot)\) and \((\mathbb{R}, \cdot)\)?

- **Discussion:** Can you turn any of them into a group easily?

- Let \(\mathbb{Z}_n = \{0, 1, \ldots, n - 1\}\) (the set of remainders *modulo* \(n\)). Define

\[
\forall a, b \in \mathbb{Z}_n : a + b = (a + b) \mod n \quad \text{and} \quad a \cdot b = (a \cdot b) \mod n.
\]

- Is \((\mathbb{Z}_n, +)\) a group? What about \((\mathbb{Z}_n, \cdot)\)?
Groups: more examples

- Discussion: Is \((\mathbb{Z}, \cdot)\) a group? What about \((\mathbb{Q}, \cdot)\) and \((\mathbb{R}, \cdot)\)?
- Discussion: Can you turn any of them into a group easily?
- Let \(\mathbb{Z}_n = \{0, 1, \ldots, n - 1\}\) (the set of remainders modulo \(n\)). Define

\[
\forall a, b \in \mathbb{Z}_n : a + b = (a + b) \mod n \text{ and } a \cdot b = (a \cdot b) \mod n.
\]

- Is \((\mathbb{Z}_n, +)\) a group? What about \((\mathbb{Z}_n, \cdot)\)?
- Discussion: Under which condition is \((\mathbb{Z}_n \setminus \{0\}, \cdot)\) a group?
Groups: more examples

- **Discussion:** Is \((\mathbb{Z}, \cdot)\) a group? What about \((\mathbb{Q}, \cdot)\) and \((\mathbb{R}, \cdot)\)?
- **Discussion:** Can you turn any of them into a group easily?
- Let \(\mathbb{Z}_n = \{0, 1, \ldots, n - 1\}\) (the set of remainders \(mod\)ulo \(n\)). Define

\[
\forall a, b \in \mathbb{Z}_n : a + b = (a + b) \mod n \text{ and } a \cdot b = (a \cdot b) \mod n .
\]

- Is \((\mathbb{Z}_n, +)\) a group? What about \((\mathbb{Z}_n, \cdot)\)?
- **Discussion:** Under which condition is \((\mathbb{Z}_n \setminus \{0\}, \cdot)\) a group?
- For a prime \(p\), we denote the \((\mathbb{Z}_p \setminus \{0\}, \cdot)\) also as \(\mathbb{Z}_p^*\).
Groups: more examples

- **Discussion:** Is \((\mathbb{Z}, \cdot)\) a group? What about \((\mathbb{Q}, \cdot)\) and \((\mathbb{R}, \cdot)\)?
- **Discussion:** Can you turn any of them into a group easily?
- Let \(\mathbb{Z}_n = \{0, 1, \ldots, n - 1\}\) (the set of remainders *modulo* \(n\)). Define

\[
\forall a, b \in \mathbb{Z}_n : a + b = (a + b) \mod n \quad \text{and} \quad a \cdot b = (a \cdot b) \mod n.
\]

- Is \((\mathbb{Z}_n, +)\) a group? What about \((\mathbb{Z}_n, \cdot)\)?
- **Discussion:** Under which condition is \((\mathbb{Z}_n \setminus \{0\}, \cdot)\) a group?
- For a prime \(p\), we denote the \((\mathbb{Z}_p \setminus \{0\}, \cdot)\) also as \(\mathbb{Z}_p^*\).
- **Discussion:** Prove that multiplication table of a (finite) group is a Latin square, i.e. each element occurs in each row and column exactly once.
Groups: integer * element (element^integer)

- For \( a \in G \) and \( n \in \mathbb{Z} \) denote

\[
\underbrace{a \oplus a \oplus \ldots \oplus a}_{n \text{ times}} \overset{\text{def}}{=} n \ast a.
\]

- If \( n = -1 \) then \((-1) \ast a = -a\), inverse of \( a \).
- In general, \((-n) \ast a = -(n \ast a)\).
- In multiplicative notation, we write \( a^n \), \( a^{-1} \), etc.
Cyclic groups, generators, order

- Let \( g \in G \) and consider the set
  \[
  \langle g \rangle = \{ i \ast g : i \in \mathbb{Z} \} = \{ \ldots, (-2) \ast g, -g, e, g, 2 \ast g, \ldots \}.
  \]

- If \( G = \langle g \rangle \), we say that the group \( G \) is cyclic and generated by \( g \) (or \( g \) is a generator of \( G \)).

- All cyclic groups are isomorphic to either \( \mathbb{Z} \) or \( \mathbb{Z}_n \) for some \( n \).

- If for \( g \in G \) there exists a number \( k > 0 \) such that \( k \ast g = e \), the smallest such \( k \) is called the order of the element \( g \) and denoted \( \text{ord}(g) \).

- For a finite cyclic group \( G = \langle g \rangle \), we have \( \text{ord}(g) = |G| \).

- **Discussion:** Find all the generators of \((\mathbb{Z}_{11}, +)\) and \( \mathbb{Z}_{11}^* \).
  - Useful fact: for every prime \( p \), \( \mathbb{Z}_p^* \) is cyclic.
Cyclic groups, generators, order

- Let \( g \in G \) and consider the set
  \[
  \langle g \rangle = \{ i \ast g : i \in \mathbb{Z} \} = \{ \ldots, (-2) \ast g, -g, e, g, 2 \ast g, \ldots \} .
  \]
- If \( G = \langle g \rangle \), we say that the group \( G \) is cyclic and generated by \( g \) (or \( g \) is a generator of \( G \)).
- All cyclic groups are isomorphic to either \( \mathbb{Z} \) or \( \mathbb{Z}_n \) for some \( n \).
- If for \( g \in G \) there exists a number \( k > 0 \) such that \( k \ast g = e \), the smallest such \( k \) is called the order of the element \( g \) and denoted \( \text{ord}(g) \).
- For a finite cyclic group \( G = \langle g \rangle \), we have \( \text{ord}(g) = |G| \).
- **Discussion:** Find all the generators of \((\mathbb{Z}_{11}, +)\) and \( \mathbb{Z}^*_{11} \).
  - Useful fact: for every prime \( p \), \( \mathbb{Z}^*_p \) is cyclic.
- **Discussion:** Find a non-cyclic infinite and a non-cyclic finite group.
Fundamental Theorem

Let \((G, \otimes)\) be a finite group. Let \(g \leftarrow G\) have arbitrary distribution and let \(h \leftarrow \overset{U}{G}\) have uniform distribution in \(G\). Then \(g \otimes h\) has uniform distribution in \(G\).

Discussion: prove the Theorem.
**Fundamental Theorem**

**Theorem**

Let \((G, \bigodot)\) be a finite group. Let \(g \leftarrow G\) have arbitrary distribution and let \(h \leftarrow G\) have uniform distribution in \(G\). Then \(g \bigodot h\) has uniform distribution in \(G\).

- **Discussion:** prove the Theorem.
- **Hint:** Use the Latin square property we proved before.
Encrypting one bit

In general, an encryption scheme provides algorithms

\[ \text{Enc} : \mathcal{M} \times \mathcal{K} \rightarrow \mathcal{C} \quad \text{and} \quad \text{Dec} : \mathcal{C} \times \mathcal{K} \rightarrow \mathcal{M} \]

so that

\[ \forall m \in \mathcal{M} \forall k \in \mathcal{K} : \text{Dec}(\text{Enc}(m, k), k) = m . \]
Encrypting one bit

- In general, an encryption scheme provides algorithms
  $Enc : \mathcal{M} \times \mathcal{K} \to \mathcal{C}$ and $Dec : \mathcal{C} \times \mathcal{K} \to \mathcal{M}$ so that
  $$\forall m \in \mathcal{M} \forall k \in \mathcal{K} : Dec(Enc(m, k), k) = m .$$

- Consider the group $(\mathbb{Z}_2, +)$ with the addition table
  
  $\begin{array}{c|cc}
    + & 0 & 1 \\
    \hline
    0 & 0 & 1 \\
    1 & 1 & 0 \\
  \end{array}$

- Let the message (plaintext) space $\mathcal{M} = \mathbb{Z}_2$, key space $\mathcal{K} = \mathbb{Z}_2$ and ciphertext space $\mathcal{C} = \mathbb{Z}_2$.

- Let the encryption be defined as
  $$Enc(m, k) = m + k \text{ mod } 2 \quad (= m \oplus k = m \text{ XOR } k) .$$

- **Discussion:** How does $Dec$ work?
One-Time Pad

- **Discussion:** How to encrypt more bits?
One-Time Pad

- **Discussion:** How to encrypt more bits?
- Use many one-bit encryptions in parallel!
- Let $\mathcal{M} = \mathcal{K} = \mathcal{C} = \mathbb{Z}_2^n$, and let $Enc = Dec = \oplus$.

- **Discussion:** If $k \xleftarrow{U} \mathbb{Z}_2^n$ is selected uniformly randomly, the plaintext and ciphertext are completely uncorrelated. In this case we say that the scheme provides **perfect secrecy**.
One-Time Pad

- **Discussion:** How to encrypt more bits?
- Use many one-bit encryptions in parallel!
- Let $M = K = C = \mathbb{Z}_2^n$, and let $Enc = Dec = \oplus$.

- **Discussion:** If $k \xleftarrow{\text{U}} \mathbb{Z}_2^n$ is selected uniformly randomly, the plaintext and ciphertext are completely uncorrelated. In this case we say that the scheme provides **perfect secrecy**.
- **Discussion:** What happens if the key $k$ is re-used?
Shannon’s theorem

Theorem (Claude Shannon, 1940s)
For every perfectly secure cipher \((Enc, Dec)\) with message space \(\mathcal{M}\) and key space \(\mathcal{K}\), it holds that \(|\mathcal{K}| \geq |\mathcal{M}|\).

Discussion: What does this mean in terms of key length vs. message length? And in turn, what does this mean from the viewpoint of key management?
Making key management efficient

- **Key idea:**
  - let \( k \) be a short seed (say, a few hundred bits), and
  - find a way to extend it into a long random-looking bitstring;
  - finally, use this bitstring as the key in One-Time Pad.
- There are several ways of implementing this idea, we will present one based on block ciphers.
For a given block size $b$ (say, $b = 128$ bits), and key size $s$ (say, $s = 256$ bits), block cipher encryption and decryption are transformations

$$\mathbb{Z}_2^b \times \mathbb{Z}_2^s \rightarrow \mathbb{Z}_2^b .$$

For any given key $k$, the corresponding mapping

$$Enc_k : \mathbb{Z}_2^b \rightarrow \mathbb{Z}_2^b$$

should be indistinguishable from a random permutation.

Some block ciphers: AES, 3DES, Camellia, Blowfish.
**Block cipher in Counter Mode**

**Discussion:** How does CTR-mode decryption work?
CTR mode security

Discussion: Is AES-CTR secure assuming AES is a strong pseudorandom permutation?
CTR mode security

- **Discussion:** Is AES-CTR secure assuming AES is a strong pseudorandom permutation?
- **Discussion:** Is electronic voting secure?
- **Discussion:** Is an airplane secure?
CTR mode security

- **Discussion**: Is AES-CTR secure assuming AES is a strong pseudorandom permutation?
- **Discussion**: Is electronic voting secure?
- **Discussion**: Is an airplane secure?
- The key to answering such questions is to *define* what do we mean by “being secure”.
- AES-CTR protects *privacy*, but not *integrity* of the message.
- **Discussion**: Describe integrity attacks against AES-CTR and discuss their implications in practice.
CTR mode security

- **Discussion:** Is AES-CTR secure assuming AES is a strong pseudorandom permutation?
- **Discussion:** Is electronic voting secure?
- **Discussion:** Is an airplane secure?

The key to answering such questions is to define what do we mean by “being secure”.

AES-CTR protects privacy, but not integrity of the message.

- **Discussion:** Describe integrity attacks against AES-CTR and discuss their implications in practice.

In order to also provide integrity, a message authentication code/tag (MAC) can be added.
Galois Counter Mode (GCM)
Security of AES-GCM

Discussion: Is AES-GCM now secure?
Security of AES-GCM

- **Discussion:** Is AES-GCM now secure?
- Good that you asked “In what sense?” :)
- Does it provide privacy and integrity assuming that AES is a strong pseudorandom permutation?
Security of AES-GCM

- **Discussion:** Is AES-GCM now secure?
- Good that you asked “In what sense?” :)
- Does it provide privacy and integrity assuming that AES is a strong pseudorandom permutation?
  - Not automatically. If you reuse the IV/nonce, privacy can be severely harmed.
- **Discussion:** Find a privacy problem if two messages are encrypted using the same IV/nonce.
Security of AES-GCM

**Discussion:** Is AES-GCM now secure?
- Good that you asked “In what sense?” :)
- Does it provide privacy and integrity assuming that AES is a strong pseudorandom permutation?
- Not automatically. If you reuse the IV/nonce, privacy can be severely harmed.

**Discussion:** Find a privacy problem if two messages are encrypted using the same IV/nonce.
- But if used correctly, AES-GCM and other authenticated encryption modes (e.g. CCM) are fine. In TLS 1.3, un-authenticated modes are not even supported.
What about other groups?

- This far, we have been only using the groups \((\mathbb{Z}_2, +)\), but the Fundamental Theorem is far more general.
- Consider a general group \((G, \cdot)\).
- In order to protect message \(m \in G\), we can select a secret key \(s \in G\) and send \(m \cdot s\) over the communication channel.
- **Discussion:** What are the shortcomings of this scheme?
What about other groups?

- This far, we have been only using the groups \((\mathbb{Z}_2^b, +)\), but the Fundamental Theorem is far more general.
- Consider a general group \((G, \cdot)\).
- In order to protect message \(m \in G\), we can select a secret key \(s \in G\) and send \(m \cdot s\) over the communication channel.

**Discussion:** What are the shortcomings of this scheme?

- Again, key management is a major problem.
  - You can use every \(s\) only once. (Why?)
  - How do you get \(s\) over to the receiver’s side?
Protecting the secret key

- To generate a new key every time, fix an element $y \in G$ and select a fresh random $k \in \{1, \ldots, \text{ord}(G) - 1\}$. Let $s = y^k$.
- With hand-waving, this exponent can be computed efficiently.
- OK, but how do you get all the keys over to the receiver?
Protecting the secret key

- To generate a new key every time, fix an element \( y \in G \) and select a fresh random \( k \in \{1, \ldots, \text{ord}(G) - 1\} \). Let \( s = y^k \).
- With hand-waving, this exponent can be computed efficiently.
- OK, but how do you get all the keys over to the receiver?
- Select \( y \) in a clever way and give the receiver a trap-door!
- Let \( g \) be a public element of a large order in \( G \) (say, a generator). Select \( a \in \{1, \ldots, \text{ord}(G) - 1\} \) and let \( y = g^a \). Give \( a \) to the receiver as a long-term secret key (trap-door).
- Let the ciphertext of \( m \) be \( (m \cdot y^k, g^k) \).
- This scheme is known as ElGamal encryption.
- Discussion: how does the receiver decrypt the ciphertext?
Security of ElGamal encryption

- ElGamal scheme is an example of public key encryption.
  - $G$ and $g$ are agreed upon openly (they can e.g. be standardised).
  - The value $a$ is the long-term private key of the receiver.
  - The value $y = g^a$ is made public as belonging to the receiver.

**Discussion:** what properties need to be satisfied in order for ElGamal encryption to be secure?
Security of ElGamal encryption

- ElGamal scheme is an example of **public key encryption**.
  - $G$ and $g$ are agreed upon openly (they can e.g. be standardised).
  - The value $a$ is the long-term private key of the receiver.
  - The value $y = g^a$ is made public as belonging to the receiver.

**Discussion:** what properties need to be satisfied in order for ElGamal encryption to be secure?

- Glad you asked “In which sense?” again :)
- What do we need for **confidentiality**, i.e. that no-one else besides the intended recipient could read the message?
- What do we need for **authenticity**, i.e. to make sure that the public key $y$ really belongs to the claimed recipient?
Discrete logarithm problem

- To have message confidentiality, an attacker should not be able to compute the secret value $a$ from public parameters $g$ and $y = g^a$.
- That is, it should be hard to compute (discrete) logarithms in the group $G$.
- This property can be achieved by carefully selecting the group.
- One popular choice is $(\mathbb{Z}_p^*, \cdot)$ for a large prime $p$.
- For example, Estonian Internet voting uses ElGamal encryption modulo

$$p = 2^{3072} - 2^{3008} - 1 + 2^{64} \cdot \{2^{2942} \pi\} + 1690314.$$
Consider a group $G$ and an element $g \in G$. Let $a \leftarrow \{0, 1, \ldots, \text{ord}(G) - 1\}$ be uniformly distributed. Is $g^a$ a uniformly distributed element of $G$? If yes, justify; if no, find a condition on $g$ when it is so (and of course, justify as well).
More on authenticity

- Consider the following typical scenario of public key encryption.
  - Alice wants to send message $m$ to Bob over a public network.
  - She obtains Bob’s public key $pk_{Bob}$ and encrypts $Enc_{pk_{Bob}}(m)$.

Discussion: What happens if $pk_{Bob}$ isn’t really Bob’s public key and someone else has access to the corresponding private key?
More on authenticity

- Consider the following typical scenario of public key encryption.
  - Alice wants to send message $m$ to Bob over a public network.
  - She obtains Bob’s public key $pk_{Bob}$ and encrypts $Enc_{pk_{Bob}}(m)$.

- **Discussion:** What happens if $pk_{Bob}$ isn’t really Bob’s public key and someone else has accesss to the corresponding private key?

- **Discussion:** We can use a symmetric MAC for authenticity. What are the shortcomings of such an approach in the public key encryption scenario?
Digital signatures

- We need a new digital signature primitive $\text{Sig}_{\text{Bob}}$ such that
  - only Bob can create $\text{Sig}_{\text{Bob}}(m)$, but
  - everyone can verify that $\text{Sig}_{\text{Bob}}(m)$ is Bob’s signature on $m$.

**Discussion:** How can such a primitive be used to solve the problem on the last slide?
Digital signatures

- We need a new digital signature primitive $\text{Sig}_{Bob}$ such that
  - only Bob can create $\text{Sig}_{Bob}(m)$, but
  - everyone can verify that $\text{Sig}_{Bob}(m)$ is Bob’s signature on $m$.

**Discussion:** How can such a primitive be used to solve the problem on the last slide?

- In addition, to get a legal equivalent of a hand-written signature, we also require *non-repudiation*, i.e. it should be impossible for Bob to deny having signed the document.
Digital signatures

- We need a new **digital signature** primitive $\text{Sig}_{Bob}$ such that
  - only Bob can create $\text{Sig}_{Bob}(m)$, but
  - everyone can verify that $\text{Sig}_{Bob}(m)$ is Bob’s signature on $m$.

- **Discussion:** How can such a primitive be used to solve the problem on the last slide?

- In addition, to get a legal equivalent of a hand-written signature, we also require **non-repudiation**, i.e. it should be impossible for Bob to deny having signed the document.

- How can we implement such a primitive? Again, using public key cryptography!
Theorem

Let $p$ and $q$ be two different odd primes and $n = p \cdot q$. Let
\[ \varphi(n) = |\mathbb{Z}_n^*| = (p - 1) \cdot (q - 1). \]
Choose $d \in \mathbb{Z}_{\varphi(n)}^*$ and $e = d^{-1} \in \mathbb{Z}_{\varphi(n)}^*$. Then
\[ \forall m \in \mathbb{Z}_n : (m^d)^e = m \in \mathbb{Z}_n. \] (1)
RSA digital signature scheme

- Rivest-Shamir-Adleman theorem allows us to create an analogue to a hand-written signature:
  - $p$ and $q$ are secret primes needed to set up the scheme,
  - $n = p \cdot q$ is a public modulus,
  - $m$ is the message to be signed,
  - $d$ is a (private) signature key,
  - $e$ is a (public) verification key,
  - $m^d$ is a signature of the message $m$,
  - To verify the signature, check the identity (1).

- Having the verification key, document and signature, the signature key can not be found.
RSA digital signature scheme

- Rivest-Shamir-Adleman theorem allows us to create an analogue to a hand-written signature:
  - $p$ and $q$ are secret primes needed to set up the scheme,
  - $n = p \cdot q$ is a public modulus,
  - $m$ is the message to be signed,
  - $d$ is a (private) signature key,
  - $e$ is a (public) verification key,
  - $m^d$ is a signature of the message $m$,
  - To verify the signature, check the identity (1).

- Having the verification key, document and signature, the signature key can not be found.

- **Discussion:** RSA scheme can also be used for public key encryption. How?
Hybrid schemes

- The problem with public key encryption is that it is quite limited.
- The size of the messages to be encrypted is a few thousand bits.
- On a regular hardware, you can perform a few hundred decryptions per second.
- **Discussion:** RSA encryption is usually much faster than decryption. Why?

- In order to encrypt long messages fast, *hybrid encryption* is usually deployed.
- In order to send a long message $m$ to Bob, Alice first creates a symmetric (say, AES-GCM256) key $k$.
- She downloads Bob’s public (say, RSA) key $pk_{Bob}$ and encrypts $Enc^{RSA}_{pk_{Bob}}(k)$.
- Then she encrypts $Enc^{AES}_k(m)$ and sends both encryptions to Bob.
- **Discussion:** How does Bob decrypt?
Key exchange problem

- Consider the following protocol for sending an encrypted message $m$ from Alice to Bob.
  - $A$: generate a random symmetric key $K \in \mathbb{Z}_2^n$
  - $A \rightarrow B$: $K$
  - $A \rightarrow B$: $Enc_K(m)$

**Discussion:** What is a problem with this protocol?
Key exchange problem

- Consider the following protocol for sending an encrypted message $m$ from Alice to Bob.
  - A: generate a random symmetric key $K \in \mathbb{Z}_2^n$
  - A $\rightarrow$ B: $K$
  - A $\rightarrow$ B: $Enc_K(m)$

- **Discussion:** What is a problem with this protocol?

- But how do you solve this problem? In order to initiate a secure message exchange, Alice and Bob would need a secure channel to exchange the key first . . .
Key exchange problem

- Consider the following protocol for sending an encrypted message $m$ from Alice to Bob.
  - $A$: generate a random symmetric key $K \in \mathbb{Z}_2^n$
  - $A \rightarrow B$: $K$
  - $A \rightarrow B$: $Enc_K(m)$

**Discussion:** What is a problem with this protocol?

- But how do you solve this problem? In order to initiate a secure message exchange, Alice and Bob would need a secure channel to exchange the key first . . .

- It is this word ”secure” again! In order to set up a key for secret communication, it is enough to have access to an authenticated channel only!
Diffie-Hellman key exchange
Diffie-Hellman key exchange

$GF(p), g \in \mathbb{Z}_p$
Diffie-Hellman key exchange

$GF(p), \ g \in \mathbb{Z}_p$

$a \in \mathbb{Z}$
Diffie-Hellman key exchange

$GF(p), g \in \mathbb{Z}_p$

$a \in \mathbb{Z}$

$b \in \mathbb{Z}$
Diffie-Hellman key exchange

$GF(p), g \in \mathbb{Z}_p$

$a \in \mathbb{Z}$

$b \in \mathbb{Z}$
Diffie-Hellman key exchange

$GF(p), g \in \mathbb{Z}_p$

$a \in \mathbb{Z}$

$b \in \mathbb{Z}$
Diffie-Hellman key exchange

$GF(p), g \in \mathbb{Z}_p$

$a \in \mathbb{Z}$

$b \in \mathbb{Z}$

$(g^b)^a = (g^a)^b$
Authenticated key exchange

- **Discussion:** Is Diffie-Hellman key exchange secure?
Authenticated key exchange

- **Discussion:** Is Diffie-Hellman key exchange secure?
  - Right, in which sense?
  - It is not *authenticated*, i.e. in the end of the protocol Alice knows she has a key shared with someone, but is it really Bob?
  - E.g. there can be a man-in-the-middle who runs the DH protocol with both Alice and Bob and will re-encrypt all their subsequent communication, reading it all in clear text in between.
  - In order to counter this attack, one option is to *sign* the values $g^a$ and $g^b$ by Alice and Bob, respectively.
We continue with the set-up of assignment #1. Under the condition when $g^a$ and $g^b$ are uniformly distributed, is this also true for $g^{ab}$?
Confidentiality vs privacy

- Does encryption provide privacy?
Confidentiality vs privacy

- Does encryption provide privacy?
- Not necessarily. Besides the *content* of the message you may want to hide other things as well, like
  - who sent the message,
  - to whom the message was sent,
  - or that the message/data even existed at all.
Confidentiality vs privacy

- Does encryption provide privacy?
- Not necessarily. Besides the *content* of the message you may want to hide other things as well, like:
  - who sent the message,
  - to whom the message was sent,
  - or that the message/data even existed at all.

**Discussion:** find a scenario for each of the above points.
Confidentiality vs privacy

- Does encryption provide privacy?
- Not necessarily. Besides the *content* of the message you may want to hide other things as well, like
  - who sent the message,
  - to whom the message was sent,
  - or that the message/data even existed at all.

**Discussion:** find a scenario for each of the above points.

**Discussion:** find other properties of data that encryption does not protect.
Deniable encryption

- Proposed by Canetti et al. in 1997.
- Comes in flavours:
  - sender-deniable, receiver-deniable, bideniable;
  - public-key and shared-key versions.
- Let $c = E(m, k)$ be a ciphertext of message $m$ under key $k$. The idea is to open $c$ to a fake message $m_f$ by presenting a fake key $k_f$ so that $c = E(m_f, k_f)$.
- It may be necessary to fix $m_f$ already during the first encryption process.
Simple shared-key bideniable scheme

- **Discussion:** build one.
Simple shared-key bideniable scheme

- **Discussion:** build one.
- **Hint:** Use one-time-pad.
Shared-key receiver-deniable scheme

- Alice and Bob will use ElGamal cryptosystem working over $GF(p^r)$ with the multiplicative group $G = \langle g \rangle$.
- Private key is $x \in [2, 3, \ldots, \text{ord}(g) - 1]$, public key is $y = g^x$.
- Reminder: to encrypt a message $M \in G$, Alice generates a random $k \in [2, 3, \ldots, \text{ord}(g) - 1]$ and computes the cryptogram $(\alpha, \beta)$, where

$$\alpha = g^k \quad \text{and} \quad \beta = M \cdot y^k.$$ 

- To decrypt, Bob computes

$$\beta \cdot \alpha^{-x} = (M \cdot y^k) \cdot g^{-kx} = M.$$
Shared-key receiver-deniable scheme

- For deniability, Alice and Bob also pre-exchange a subliminal shared secret $s$ and a public hash function $H$.
- To encrypt a message $m_f$ and an illegal message $m \in G$, a number $k = H(s || m_f)$ is computed.
- Then Alice computes

$$\alpha = g^k \cdot m \quad \text{and} \quad \beta = (y^k \cdot m^x) \cdot m_f.$$
Shared-key receiver-deniable scheme

- For deniability, Alice and Bob also pre-exchange a subliminal shared secret $s$ and a public hash function $H$.
- To encrypt a message $m_f$ and an illegal message $m \in G$, a number $k = H(s||m_f)$ is computed.
- Then Alice computes

$$\alpha = g^k \cdot m \quad \text{and} \quad \beta = (y^k \cdot m^x) \cdot m_f .$$

**Discussion:** Prove that the pair $(\alpha, \beta)$ is a regular ElGamal ciphertext of $m_f$.

**Discussion:** How can Bob decrypt the message $m$?

**Discussion:** Would this scheme work as an addition for the Estonian e-voting system?
Electronic voting: case study 1

- Problem: ballot box stuffing (adding ineligible/unauthorised votes)
- Solution: ballot stamping

Blind digital stamping?

- To protect vote confidentiality, the stamping authority should not see what it stamps.
- Yet, when looking at the vote later, authority’s signature should be verifiable.
Blind RSA signatures

- Let the authority’s public RSA key be \((e, n)\) and secret key \(d\).
- To authorise the vote \(v\), the voter first generates random \(r \leftarrow \mathbb{Z}_n\) and sets the blinded vote \(v' = (v \cdot r^e) \mod n\).
- The voter sends \(v'\) for signing to the authority.
- The authority signs and returns \(s' = (v')^d \mod n\).
- The voter computes \(s = s' \cdot r^{-1} \mod n\).
Blind RSA signatures

- Let the authority’s public RSA key be \((e, n)\) and secret key \(d\).
- To authorise the vote \(v\), the voter first generates random\
  \(r \leftarrow \mathbb{Z}_n\) and sets the blinded vote \(v' = (v \cdot r^e) \mod n\).
- The voter sends \(v'\) for signing to the authority.
- The authority signs and returns \(s' = (v')^d \mod n\).
- The voter computes \(s = s' \cdot r^{-1} \mod n\).

**Discussion:** Prove that \(s\) is a valid signature of \(v\) under the authority’s key.
Homework assignment #1.3

If \( r \leftarrow \mathbb{Z}_n \) then is \( (r^e \mod n) \) a uniformly random element of \( \mathbb{Z}_n \)? What about \( (v \cdot r^e) \mod n \)?
RSA can also be used as a public key encryption scheme where the cryptogram of message $m$ under the public key $(e, n)$ is computed as

$$c = m^e \mod n$$

and decryption with the private key $d$ works as

$$c^d \mod n.$$

E.g., Estonian Internet voting scheme used RSA encryption before 2017.

What would happen if the same RSA key pair would be used in a voting scheme both for encryption and blindly signing the votes?
Untraceable electronic cash

- We want a scheme featuring a bank, a client and a shop.
- The bank should be able to issue digital coins that the client could spend in the shop so that
  - the shop can check that the coin was issued by the bank,
  - the shop and the bank can not reveal the identity of the client even if they cooperate, and
  - no coin could be doubly spent.
- **Discussion:** Build such a system.
- **Hint:** use blind signatures.
A simple authentication protocol

Stop! Your password!

Password

OK, you may pass!

Discussion: What is wrong with this protocol (even assuming the communication is encrypted)?
A simple authentication protocol

Stop! Your password!

Password

OK, you may pass!

Discussion: What is wrong with this protocol (even assuming the communication is encrypted)?

- The Verifier obtains full credentials to impose as the Prover!
Schnorr authentication protocol

- The 🎅 selects a group $G$ of order $q$ where DL is hard, picks a generator $g$ and secret $a \in \mathbb{Z}_q$, and publishes $y = g^a$.
- Authentication is achieved by demonstrating that 🎅 knows $a$.

\[
\begin{align*}
  k & \leftarrow \mathbb{Z}_q \\
  h &= g^k \\
  c & \leftarrow \mathbb{Z}_q \\
  s &= ac + k \mod q \\
  \text{Check } g^s &= y^c \cdot h
\end{align*}
\]
Schnorr authentication protocol

- The Santa selects a group $G$ of order $q$ where DL is hard, picks a generator $g$ and secret $a \in \mathbb{Z}_q$, and publishes $y = g^a$.

- Authentication is achieved by demonstrating that Santa knows $a$.

\[
\begin{align*}
  k & \leftarrow \mathbb{Z}_q \\
  h &= g^k \\
  c & \leftarrow \mathbb{Z}_q \\
  s &= ac + k \mod q \\
  \text{Check } g^s &= y^c \cdot h
\end{align*}
\]

Discussion: Is Schnorr authentication protocol secure?
Zero-knowledge proofs

Definition

We say that a proof-of-knowledge protocol has zero-knowledge property if it achieves:

- **Completeness:** If the Prover knows what he claims to, then Verifier accepts the proof.
- **Soundness:** If the Verifier accepts the proof, the Prover actually knows what he claims to.
- **Zero-knowledgeness:** In the course of the proof, the Verifier learns nothing except the claim to be proven.
Schnorr: completeness

Discussion: Prove it.
Schnorr: soundness

- We will prove a somewhat stronger property – knowledge extractability.

- After the Prover has committed to $k$ by sending $g^k$, we let the Verifier make two queries with independent $c, c' \in \mathbb{Z}_q$.

  - Since formally Prover and Verifier are just Turing Machines, this is fine.

- The prover produces two values $s$ and $s'$.

- **Discussion:** Prove that from $s$ and $s'$ it is possible (with high probability) to extract $a$. 
Schnorr: zero-knowledgeness

- What does it actually mean that the Verifier learns *nothing* beyond the fact that the Prover knows $a$?
What does it actually mean that the Verifier learns *nothing* beyond the fact that the Prover knows $a$?

We will show that the Verifier’s *view* on the protocol can be *simulated* even without having access to $a$ at all!

Verifier’s view is:

1. A uniform group element $h$ comes in.
2. I generate and send back $c \leftarrow \mathbb{Z}_q$ that is *independent* of $h$.
3. A uniform element $s \in \mathbb{Z}_q$ comes in.
4. I accept if $g^s = y^c \cdot h$. 
Simulating Verifier’s view

- Recall: Verifier is just a Turing Machine than can be run again (rewinded) from some previous state.
- We let the Verifier run for two first steps to see which \( c \) it produces.
- Then we start the Verifier again by first generating a fresh \( r \leftarrow \mathbb{Z}_q \) and feeding the Verifier with \( g^r \cdot y^{-c} \) on step 1 and with \( r \) on step 3.
Simulating Verifier’s view

- Recall: Verifier is just a Turing Machine than can be run again (rewinded) from some previous state.
- We let the Verifier run for two first steps to see which $c$ it produces.
- Then we start the Verifier again by first generating a fresh $r \leftarrow \mathbb{Z}_q$ and feeding the Verifier with $g^r \cdot y^{-c}$ on step 1 and with $r$ on step 3.
- **Discussion:** Prove that 1) the Verifier accepts these values, and 2) the distribution of the Verifier’s view is exactly as above.
Electronic voting: case study 2

- Problem: when the votes are decrypted for tallying, the decryption server cannot reveal the private key.
  - **Discussion:** Why?
- But then, how to prove that the decryption server acted honestly?
- **Solution:** proofs proving that the decryption was correct *and nothing else* (i.e. zero-knowledge proofs).
Estonian i-voting ElGamal encryption

- Estonian Internet voting uses 3072-bit MODP ElGamal/DH group from RFC 3526 with generator $g = 2$ of order $q$.
- There is a (distributed) decryption key $x$ and the corresponding public key $h = g^x$.
- To encrypt a vote $m$, choose $R \leftarrow \mathbb{Z}_q$ and let the corresponding ElGamal cryptogram be
  $$(c_1, c_2) = (m \cdot h^R, g^R).$$
- With the decryption key $x$, the decryption server computes
  $$d = \frac{c_1}{c_2^x}. \quad (2)$$
- **Discussion:** Prove that if the decryption server behaves correctly, we get $d = m$. 

March 2020
Creating the decryption proof

- Equation (2) is equivalent to

\[ \frac{c_1}{d} = c_2^x \]  

(3)

where \( x \) is the private exponent.

- Essentially, we will be using Schnorr protocol to prove the knowledge of \( x \).

- The decryption server (prover) picks \( r \leftarrow \mathbb{Z}_q \) and computes two commitments

\[ a = c_2^r \quad \text{and} \quad b = g^r . \]

- Verifier generates a challenge \( k \in \mathbb{Z}_q \) and presents it to the prover.

- The prover computes \( s = kx + r \mod q \) and presents \((a, b, s)\) as a proof.
Verifying the decryption proof

- **Discussion:** Prove that (3) is an implication of the (directly verifiable) equation

\[
a \cdot \left( \frac{c_1}{d} \right)^k = c_2^s.
\]

- Now \((a, s)\) certifies that the prover knows the exponent \(x\) that was used to compute \(d\) using (2). But is it the same \(x\) that corresponds to the public key \(h\)?

- This is proven by \((b, s)\). Note that

\[
b \cdot h^k = g^r \cdot g^{xk} = g^{r+xk}.
\]

Thus, it remains to verify the equation

\[
b \cdot h^k = g^s.
\]
Fiat-Shamir hack heuristic

- The problem with vanilla Schnorr protocol is that it is *interactive*, i.e. it requires the Verifier to be online.
- However, we want the proofs to be also verifiable later.
  - E.g. vote decryption should be independently auditable.
- To achieve this, Verifier’s challenge \( c \) is often replaced by a hash of all the previous values in the protocol.
  - E.g. in case of Schnorr protocol, we would take

\[
c = H(g, y, h).
\]

- Note that \( H \) must give uniform output in \( \mathbb{Z}_q \), one must take care about the modulo bias!
Homework assignment #3

☐ Recall the “Where is Waldo?” game.

☐ Design an interactive zero-knowledge proof allowing to prove that you know where Waldo is without revealing anything else about your knowledge (most notably, without showing where exactly he is).
How to leak a secret?

Who of you disclosed the UFO photos?!

Ring signatures

- There is a group (*ring*) of *r* actors $A_1, A_2, \ldots, A_r$.
- Some of them (say, $A_s$) wants to sign a message $m$ so that
  - an outside verifier could check that the signature was created by one of the ring members, but
  - the verifier could not tell who exactly.
- The other ring members do not need to participate in the signature creation. In fact, they may even be unaware that a signature has been created partly on their behalf!
- This is a sort of deniability property.
**Ring signatures: setup**

- Each ring member $A_i$ has an RSA key pair with the public key $P_i = (n_i, e_i)$ (including the signer $A_s$ who also has access to his private key $d_i$).
- The signer $A_s$ hashes the message $m$ to be signed as $k = h(m)$ and uses the result $k$ as the key for a symmetric encryption algorithm $E_k$ working on $b$-bit values (say, AES).
- $A_s$ also picks random values $x_i$ for all $A_i, i = 1, 2, \ldots, r, i \neq s$.
- $A_s$ computes $y_i = x_i^{e_i} \mod n_i$ for all $A_i, i = 1, 2, \ldots, r, i \neq s$.
  - Note that $x_i$ verifies as $A_i$’s signature on the (meaningless) message $y_i$!
- $A_s$ chooses a random initial (glue) value $v$. 
Ring signatures: how it works

\[ y_1 = x_1^{e_1} \mod n_1 \]

\[ y_2 = x_2^{e_2} \mod n_2 \]

\[ y_r = x_r^{e_r} \mod n_r \]

\[ E_k^{-1}(y_{s+1} \oplus \ldots) = E_k(y_{s-1} \oplus \ldots) \oplus y_s \]
Ring signatures: closing the ring

- $A_s$ solves the last equation for $y_s$ and computes his real signature on it: $x_s = y_s^{d_s} \mod n_s$.
- The final signature will be

$$\left( P_1, P_2, \ldots, P_r; v; x_1, x_2, \ldots, x_r \right)$$

- **Discussion:** How does one verify this signature?
- **Discussion:** Why does this construction achieve all the required properties of a ring signature?
Based on the above-described ring signature scheme, build a designated verifier signature scheme. That is,

- Alice wants to sign a message $m$ so that only Bob would be convinced by the signature.
- For any other verifier, it looks exactly as Bob could have generated the signature himself.
Distributing encryption keys

What is your public key?

$m$

Discussion: What problems may occur here?
Distributing encryption keys

What is your public key?

Discussion: What problems may occur here?

- Bob may be offline when Alice sends the message, or may otherwise be unaware that Alice wants to send him something.
- So we would like to have a mechanism such that Alice can send Bob a message based on his public *non-cryptographic* ID only (e.g. email address).
Basic scheme

1. $bob@example.com$

2. $pk_B$

3. $Enc_{pk_B}(m)$

4. $sk_B$
Public ID → public key, attempt 1

- Let’s use ElGamal encryption, then public key $y$ is just a group element.
- Discussion: Can we take a hash function $H$ that produces elements of the ElGamal group $G$ and set

$$y = H(bob@example.com)$$
Let’s use ElGamal encryption, then public key \( y \) is just a group element.

**Discussion:** Can we take a hash function \( H \) that produces elements of the ElGamal group \( G \) and set

\[
y = H(\text{bob@example.com})?
\]

The problem is that Bob should also know \( a \) such that \( y = g^a \), i.e. a discrete logarithm of \( y \), which should be infeasible.
Public ID → public key, attempt 2

- Assume we have a magical function $D : G \times G \rightarrow G$ such that given two group elements $H(id) = g^a$ and a master public key $pk = g^s$, somehow compute the combined value $g^{as}$.
- Then we could encrypt under both the hash value $h = g^a$ and the public key $pk$ as follows:
  \[ C_1 = g^r, \quad C_2 = D(h^r, pk) \cdot m \quad (= g^{ras} \cdot m). \]
- To decrypt this ciphertext, you could use knowledge of $s$ and the magical function $D$ as follows:
  \[ m^l = C_2 / D(C_1, h)^s. \]

Discussion: What is a problem with this scheme?
Public ID $\rightarrow$ public key, attempt 2

- Assume we have a *magical function* $D : G \times G \rightarrow G$ such that given two group elements $H(id) = g^a$ and a master public key $pk = g^s$, somehow compute the combined value $g^{as}$.
- Then we could encrypt under *both* the hash value $h = g^a$ and the public key $pk$ as follows:

$$C_1 = g^r, C_2 = D(h^r, pk) \cdot m = (g^{ras} \cdot m).$$

- To decrypt this ciphertext, you could use knowledge of $s$ and the magical function $D$ as follows:

$$m' = C_2 / D(C_1, h)^s.$$ 

**Discussion:** What is a problem with this scheme?

- There is no such function $D$ – if there would be, it would break the Diffie-Hellman key exchange, which is equivalent of breaking ElGamal encryption.
Public ID → public key, attempt 3

- We can have something very close to the magical function $D$ if we require its output group to be $G_T \neq G$.
- We say that a map $e: G \times G \to G_T$ is bilinear if for all $a, b$ the following holds:

$$e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}.$$  

Also, we require that $e$ is not mapping everything trivially to identity.
- We let $m \in G_T$, master public key $pk = g^s$ and $h = H(bob@example.com) = g^a$ (where $a$ is not known!).
- To encrypt $m$, we generate fresh random $r$ and compute

$$C_1 = g^r, \quad C_2 = e(h^r, pk) \cdot m = e(g, g)^{ras} \cdot m.$$
Attempt 3, decryption

- With knowledge of the secret key $s$, decryption can be computed as:

$$m' = C_2 / e(C_1, h^s).$$

- **Discussion:** Is this the scheme we wanted?
Attempt 3, decryption

- With knowledge of the secret key $s$, decryption can be computed as:
  \[ m' = C_2 / e(C_1, h^s) \].

- **Discussion:** Is this the scheme we wanted?
- No, the master key $s$ (owned by the key distribution server) is used for Bob’s decryption.

- **Discussion:** How can we improve?
Attempt 3, decryption

- With knowledge of the secret key $s$, decryption can be computed as:

$$m' = C_2/e(C_1, h^s).$$

**Discussion:** Is this the scheme we wanted?

- No, the master key $s$ (owned by the key distribution server) is used for Bob’s decryption.

**Discussion:** How can we improve?

- We can give $h^s$ to Bob as his secret key!

- The result is called Boneh-Franklin identity-based encryption (IBE) scheme.
Boneh-Franklin IBE scheme

\[ msk = s, \quad mpk = g^s \]

\[ h = H(\text{bob@example.com}) \]

1. \[ C_1 = g^r, \quad C_2 = e(h^r, mpk) \cdot m \]

2. \[ h^s \]
Homework assignment #5

Prove that if $G$ is cyclic, the bilinear map $e : G \times G \to G_T$ is commutative, i.e.

$$\forall g_1, g_2 \in G : e(g_1, g_2) = e(g_2, g_1).$$
Encryption vs privacy

- In a standard public key or identity based encryption scheme, the sender knows the recipient’s identity.
- **Discussion:** Is this always what we want / need?
Encryption vs privacy

- In a standard public key or identity based encryption scheme, the sender knows the recipient’s identity.

**Discussion:** Is this always what we want / need?

**Discussion:** What is identity?
Encryption vs privacy

- In a standard public key or identity based encryption scheme, the sender knows the recipient’s identity.

**Discussion:** Is this always what we want / need?

**Discussion:** What is identity?

- A possible point of view: identity is a set of attributes.
  - Age, gender, education, organisational role, . . .

- In many situations, it is not important to fully know who the person is, but rather which attributes he/she has.
Attribute based encryption

- Say, a company CEO wants to send encrypted email to senior employees who have served the company for more than 15 years.
- Instead of manually going over the profiles of all the thousands of employees, she would like to simply state attributes during the encryption process to make sure that only the people satisfying the criteria would be able to decrypt.
- To set the scheme up, we again need a trusted party to generate and distribute attribute keys.
ABE – general setup

1. $\text{attr}_1, \text{attr}_2$

2. $\text{sk}_{\text{attr}_1}, \text{sk}_{\text{attr}_2}$

3. $\text{Enc}_{pk_{\text{attr}_i}, pk_{\text{attr}_j}}(m) \rightarrow \{\text{attr}_i, \text{attr}_j\} \subseteq \{\text{attr}_1, \text{attr}_2\}$
Sahai-Waters “Fuzzy identity” ABE

- A $d$-out-of-$n$ scheme, i.e. in order to be able to decrypt, Bob must have at least $d$ of the global $n$ parameters matching.
- Setup:
  - Let $G_1$ be bilinear group of prime order $p$, and let $g$ be a generator of $G_1$. Additionally, let $e : G_1 \times G_1 \rightarrow G_2$ denote the bilinear map.
  - Let $U = \{attr_1, attr_2, \ldots, attr_n\} = \{1, 2, \ldots, n\}$ be the set of possible attributes.
  - Also define the Lagrange polynomials $\Delta_{i,S}$ for $i \in \mathbb{Z}_p$ and a set $S$ of elements in $\mathbb{Z}_p$:

$$\Delta_{i,S}(x) = \prod_{j \in S, j \neq i} \frac{x-j}{i-j}.$$
Fuzzy identity key generation

- Master secret key is

\[ t_1, t_2, \ldots, t_n \leftarrow \mathbb{Z}_p, y \leftarrow \mathbb{Z}_p. \]

- The corresponding global public key is

\[ T_1 = g^{t_1}, T_2 = g^{t_2}, \ldots, T_n = g^{t_n}, Y = e(g, g)^y. \]

- To generate a private key for identity \( \omega \subseteq U \), first choose a \( d - 1 \) degree polynomial \( q \) is randomly chosen such that \( q(0) = y \). The private key consists of components \( (D_i)_{i \in \omega} \), where \( D_i = g^{\frac{q(i)}{t_i}} \) for every \( i \in \omega \).
Fuzzy identity encryption

To encrypt a message $M \in G_2$ for the public key $\omega' \subseteq U$,
- choose a fresh random value $s \leftarrow U$,
- compute the ciphertext as

$\left( \omega', E' = M \cdot Y^s, \{ E_i = T_i^s \}_{i \in \omega'} \right)$. 
Fuzzy identity decryption

- We will use the fact that one needs \( d \) points to reconstruct a \( d - 1 \) degree polynomial.
- If \( |\omega \cap \omega'| \geq d \), let Bob select a set \( S \subseteq \omega \cap \omega' \), \( |S| = d \).
- To decrypt the ciphertext, Bob computes

\[
E' / \prod_{i \in S} (e(D_i, E_i))^{\Delta_{i,S}(0)} = \\
= M \cdot e(g, g)^{sy} / \prod_{i \in S} \left( e\left( g^{\frac{q(i)}{t_i}}, g^{st_i} \right) \right)^{\Delta_{i,S}(0)} = \\
= M \cdot e(g, g)^{sy} / \prod_{i \in S} \left( e(g, g)^{sq(i)} \right)^{\Delta_{i,S}(0)} = \\
= M.
\]
Key-policy vs ciphertext-policy ABE

- Sahai-Waters scheme implements a threshold key-policy scheme (KP-ABE), where
  - the sender defines the attributes, and
  - the key distributor defines the access policy (in this case, \( d\)-out-of-\( n \)).

- However, sometimes the sender wants to have more control over the decryption policy. Luckily, there also exist ciphertext-policy schemes (CP-ABE), where
  - the key distributor defines the attributes, and
  - the sender can determine the policy.

- Also, it is possible to implement more complex access logic (e.g. any predicate combining AND- and OR-gates).