NTNU

Norwegian University of Science and Technology

Cryptography 101 & Secure Computation

Center for Security and Emerging Technology at Georgetown University, Washington, D.C.

Tjerand Silde

Department of Mathematical Sciences,
NTNU Trondheim

NTNU

Overview

- Cryptography 101
  - Symmetric-Key Encryption
  - Public-Key Encryption
  - Hash Functions
  - Digital Signatures
- Secure Computation
  - Secret Sharing
  - Multi-Party Computation
  - Fully Homomorphic Encryption
  - Secure computation in practice
- Cryptography & Machine Learning

NTNU

Cryptography 101

NTNU

Symmetric-Key Encryption I

A symmetric-key encryption-scheme is an encryption-system for two parties with two functions $\texttt{Enc}(\cdot,\cdot)$ and $\texttt{Dec}(\cdot,\cdot)$ together with a shared key k:

- $\texttt{Enc}$ takes as input a key k and a message m, and outputs a ciphertext c: $\texttt{Enc}(k, m) = c$.
- $\texttt{Dec}$ takes as input a key k and a ciphertext c, and outputs a message m: $\texttt{Dec}(k, c) = m$.

Only the two parties that know the key can encrypt and decrypt messages.

Symmetric-Key Encryption II

NTNU

Symmetric-Key Encryption III

Properties of a symmetric-key encryption-scheme:

- Both parties need to know the key k in advance
- Usually have keys and blocks of size 128 or 256 bits
- Super fast (only using XOR, AND, vectors etc.)
- Length of ciphertext $\approx$ Length of plaintext

NTNU

Symmetric-Key Encryption IV

Security of symmetric-key encryption:

- The best way to break it is to guess the key
- It does not hide the length of the encrypted data
- Might be vulnerable to side-channel attacks
- Might be vulnerable to attacks by quantum computers

NTNU

Public-Key Encryption I

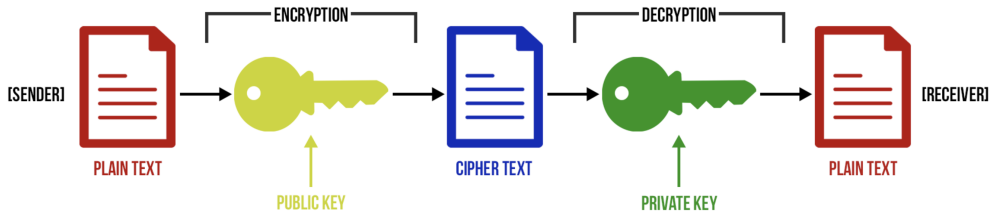
A public-key encryption-scheme is an encryption-system for two parties with two functions $\texttt{Enc}(\cdot,\cdot)$ and $\texttt{Dec}(\cdot,\cdot)$ together with a public key k_p and a secret key k_s:

— $\texttt{Enc}$ takes as input a public key k_p and a message m,
and outputs a ciphertext c: $\texttt{Enc}(k_p, m) = c$.

— $\texttt{Dec}$ takes as input a secret key k_s and a ciphertext c,
and outputs a message m: $\texttt{Dec}(k_s, c) = m$.

Everyone can encrypt a message, but only one party can decrypt.

Public-Key Encryption II



NTNU

Public-Key Encryption III

Properties of a public-key encryption-scheme:

- Only one party knows the private key, everyone knows the public key
- Pretty slow, working with big numbers thousands of bits long
- Length of ciphertext $>>$ Length of plaintext

NTNU

Public-Key Encryption IV

The RSA cryptosystem:

- Choose two prime numbers p and q and let $n = p \cdot q$
- Choose a number e and find a special number d depending on p and q
- The public key is the two numbers (e, n)
- The private key is the two numbers (d, n)

Public-Key Encryption V

The RSA cryptosystem:

- Encryption of message m: $\texttt{Enc}((e, n), m) = m^e \mod n = c$
- Decryption of ciphertext c: $\texttt{Dec}((d, n), c) = c^d \mod n = m$

Public-Key Encryption VI

Security of RSA public-key encryption:

- The best way to break RSA is to factorize n into p and q
- The cryptosystem must be randomized to be secure
- Might be vulnerable to padding attacks
- Might be vulnerable to side-channel attacks
- Will be broken by attacks by quantum computers

NTNU

Hash Functions I

A hash function `H` is a deterministic function that takes input of arbitrary length, and produce a fixed length output. A hash function has the three following properties:

- Collision resistance,
- Pre-image resistance,
- Second pre-image resistance.

Hash Functions II

Collision resistance:
It should be difficult to find two different messages m_1 and m_2 such that $\mathtt{H}(m_1) = \mathtt{H}(m_2)$.

NTNU

Hash Functions III

Pre-image resistance:
Given a hash value h it should be difficult to find any message m such that $h = \mathtt{H}(m)$.

Hash Functions IV

Second pre-image resistance:
Given an input m_1, it should be difficult to find a different input m_2 such that $\mathtt{H}(m_1) = \mathtt{H}(m_2)$.

NTNU

Hash Functions V

Security of hash-functions:

- The best way to break a hash-function is to guess inputs
- The best way to get collisions is the birthday attack
- Commonly used hash-functions have outputs of length 256 and 512 bits
- Might be vulnerable to attacks by quantum computers

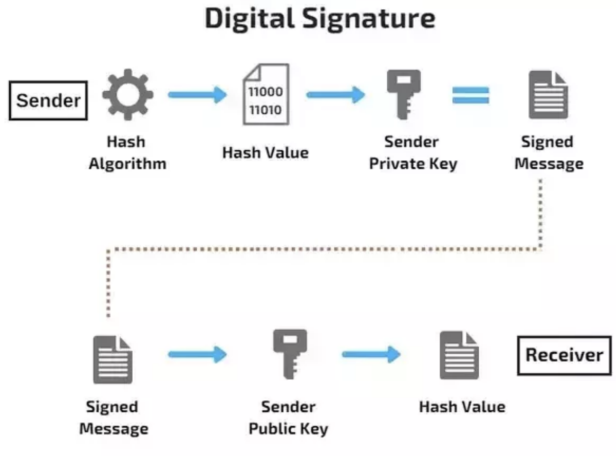
Digital Signatures I

A digital signature-scheme is a signature-system for two parties with two functions $\texttt{Sign}(\cdot,\cdot)$ and $\texttt{Verify}(\cdot,\cdot,\cdot)$ together with hash function $\texttt{H}$, a secret key k_s and a public key k_p:

- $\texttt{Sign}$ takes as input a secret key k_s and a message m, and outputs a signature σ: $\texttt{Sign}(k_s, m) = \sigma$.
- $\texttt{Verify}$ takes as input a public key k_p, a signature σ and a message m, and outputs a message **True** or **False**: $\texttt{Verify}(k_p, \sigma, m) =$ **True** or **False**.

Only one party can sign a message, but everyone can verify a signature.

Digital Signatures II



NTNU

Digital Signatures III

The RSA digital signature-system:

- Choose two prime numbers p and q and let $n = p \cdot q$
- Choose a number v and find a special number s depending on p and q
- The public key is the two numbers (v, n)
- The private key is the two numbers (s, n)

Digital Signatures IV

The RSA digital signature-system:

— Signature of message m: $\texttt{Sign}((s, n), m) = (\texttt{H}(m))^s \mod n = \sigma$

— Verification of signature σ: $\texttt{Verify}((v, n), \sigma, m) = \sigma^v \mod n \stackrel{?}{=} \texttt{H}(m)$

Digital Signatures V

Scott Hanselman
@shanselman

Follow

HTTPS & SSL doesn't mean "trust this." It means "this is private." You may be having a private conversation with Satan.

NTNU

Secure Computation

NTNU

Secure Computation

Goal:

being able to do computations on encrypted data, and therefore compute valuable information without violating privacy.

Secret Sharing I

Goal:

share a secret with a group in a way so that none of the people can reconstruct the secret on their own, but some threshold of members can reconstruct the secret if they work together.

NTNU

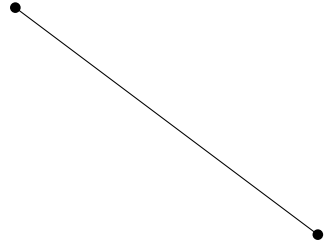
Secret Sharing II

First; some geometry:

- How many points in the plane do you need to define a straight line?
- How many points in the plane do you need to define a parabola?

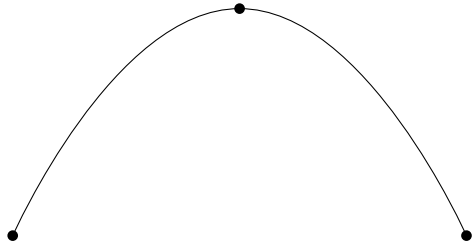
Secret Sharing III

$y = ax + b$



Secret Sharing IV

$$y = ax^2 + bx + c$$



Secret Sharing V

We have a secret number, say, $b = 4$.

We want to share this secret with three friends in a way so that none of them knows our secret, but if at least two of them work together, then they are able to reconstruct the secret.

NTNU

Secret Sharing VI

We want to construct a polynomial of degree one: $p(x) = ax + b$,
and give one point $p(1), p(2), p(3)$ to each of our three friends.

Assume that we can only chose coefficient from the numbers 0 to 4.
Our secret number will be $b = 4$, and we randomly choose, say, $a = 2$.

Secret Sharing VII

Then $p(0) = 4$ is the secret we want to share.

We share the following with our friends:

- $p(1) = 2 \cdot 1 + 4 = 6 \mod 5 = 1$ with player A,
- $p(2) = 2 \cdot 2 + 4 = 8 \mod 5 = 3$ with player B,
- $p(3) = 2 \cdot 3 + 4 = 10 \mod 5 = 0$ with player C.

Secret Sharing VIII

Individually,

- player A has the value $p(1) = 1$,
- player B has the value $p(2) = 3$,
- player C has the value $p(3) = 0$,

but none of them know what the secret is, because they need two points to be able to reconstruct the whole line.

NTNU

Multi-Party Computation I

Goal:

allow a group to compute a public function,
without having to reveal their individual input.

NTNU

Multi-Party Computation II

Procedure:

- Everyone secret-share their input with everyone
- Addition and scalar multiplication is done locally
- Multiplications require sending a new secret-share to every party

NTNU

Multi-Party Computation III

Security:

- Secret sharing with threshold $n - 1$ of n parties
- Parties can provide "false" input
- Can notice and abort if someone is cheating

NTNU

Multi-Party Computation IV

Advantages:

- Computation is really fast
- Post-quantum security

NTNU

Multi-Party Computation V

Disadvantages:

- Require a lot of communication
- Everyone must be "online" at all time

NTNU

Fully Homomorphic Encryption I

Goal:

allow a third party to compute a function,
without knowing the input nor the output.

NTNU

Fully Homomorphic Encryption II

"Noisy encryption":

- The encryption of a message contains some extra "noise"
- As long as the "noise" is small, you can still decrypt
- Adding ciphertexts also adds the "noise"
- Multiplying ciphertexts also multiplies the "noise"
- When the "noise" gets larger, you can "bootstrap" to make it smaller

Fully Homomorphic Encryption III

Procedure:

- The user encrypts their data and sends it over
- The third party does the computations on the encrypted data
- The user gets the encrypted data back and then decrypts

NTNU

Fully Homomorphic Encryption IV

Advantages:

- No communication while computing
- Small functions are fast to compute
- Post-quantum security

NTNU

Fully Homomorphic Encryption V

Disadvantages:

- Require a lot of computation
- "Bootstrapping" is expensive

NTNU

Secure computation in practice I

Multi-Party Computation:

- Pre-computation independent on data and function using FHE
- Secret share information as normal
- Fast MPC using the pre-computed data

NTNU

Secure computation in practice II

Fully Homomorphic Encryption:

- Agree on function before choosing parameters
- Choose parameters large enough to handle some "noise"
- Do as little "bootstrapping" as possible

NTNU

Secure computation in practice III

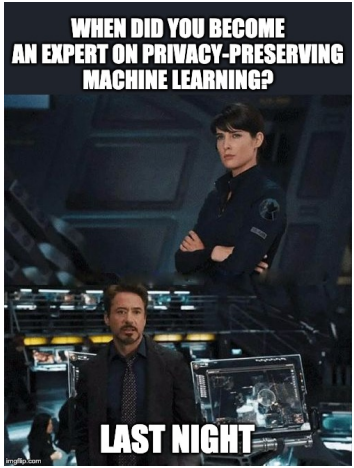
Some applications in use:

- Journalist can share sensitive information in case anything happens...
- Private companies can analyze medical information without violating privacy
- Competing companies can work together without sharing data

NTNU

Cryptography & Machine Learning

Cryptography & Machine Learning I



NTNU

Cryptography & Machine Learning II

Cryptography for machine learning:

- Training neural networks on encrypted data
- Prove correctness of machine learning algorithms
- Protect machine learning algorithms against abuse
- Training neural networks while keeping the training model secret

Cryptography & Machine Learning III

Machine learning for security:

- Detection of vulnerabilities and attacks
- Crypto- and malware-analysis
- Machine learning for attacking cryptography

Cryptography & Machine Learning IV

Resources:

— Rivest, Asiacrypt 1991. “Cryptography and machine learning”.
- https://people.csail.mit.edu/rivest/pubs/Riv91.pdf

— Goldwasser, Crypto 2018. "From Idea to Impact, the Crypto Story"
- https://www.youtube.com/watch?v=culuNbMPP0k

— Alani, 2019. "Applications of Machine Learning in Cryptography: A Survey."
- https://dl.acm.org/citation.cfm?doid=3309074.3309092

NTNU

Thank You! Questions?

Email: tjerand.silde@ntnu.no
Talk: `www.tjerandsilde.org/talks`