Computer Security Foundations
Week 10: Public Key Cryptography

Bernardo Portela

L.EIC - 24

Public Key Cryptography

Revolution in the 70s

- Before: symmetric crypto
- Pre-shared keys
- 75-78:
  - Public key encryption
  - Digital signatures
  - Key agreements

New Directions in Cryptography

Invited Paper

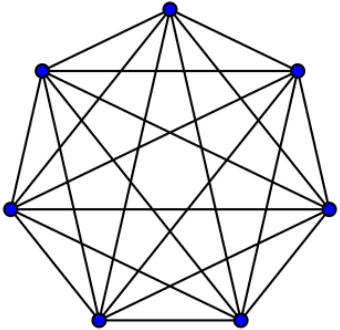
WHITFIELD DIFFIE AND MARTIN E. HELLMAN, MEMBER, IEEE

WE STAND TODAY on the brink of a revolution in cryptography. The development of cheap digital hardware has freed it from the design limitations of mechanical computing and brought the cost of high grade cryptographic devices down to where they can be used in such commercial applications as remote cash dispensers and computer terminals. In turn, such applications create a need for new types of cryptographic systems which minimize the necessity of secure key distribution channels and supply the equivalent of a written signature. At the same time, theoretical developments in information theory and computer science show promise of providing provably secure cryptosystems, changing this ancient art into a science.

Key Management

Symmetric Cryptography

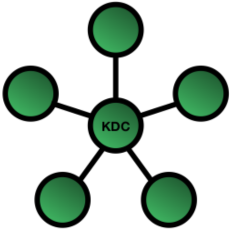
- N participants
  - Ad hoc: $\frac{n(n-1)}{2}$ keys
- Key pre-distribution
  - Manually?
  - How to dynamically change participants?



Key Management: Closed Systems

Symmetric Cryptography

- N participants
  - Centralized solution: N keys
- Key Distribution Center
  - Stores 1 long-term key, shared with each and every participant
  - How does A communicate with B?
  - Used in a generalized fashion (Kerberos)
  - Always on-line. Central point of failure



Session Keys

Modern systems distinguish

- Long-term keys $\neq$ session keys

Session Keys

Modern systems distinguish

- Long-term keys $\neq$ session keys
- Session keys: ephemeral, data limited if corrupted

Session Keys

Modern systems distinguish

- Long-term keys $\neq$ session keys
- Session keys: ephemeral, data limited if corrupted
- Long-term keys:
  - Strong security requirements in storage
  - Recall HSMs, *smartcards*, etc.

Limitations of Symmetric Cryptography

Important!

If we can use only symmetric cryptography, **good**. It's efficient and entails less complexity

Limitations of Symmetric Cryptography

Important!

If we can use only symmetric cryptography, **good**. It's efficient and entails less complexity

Problem one

- Shared long-term symmetric keys:
  - In open, asynchronous systems $\Rightarrow$ Public key encryption
  - In open, synchronous systems $\Rightarrow$ Key agreements $+$ digital signatures

Limitations of Symmetric Cryptography

Important!

If we can use only symmetric cryptography, **good**. It's efficient and entails less complexity

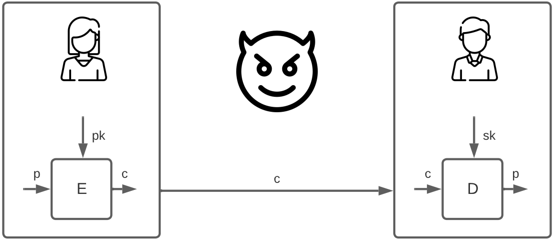
Problem one

- Shared long-term symmetric keys:
  - In open, asynchronous systems $\Rightarrow$ Public key encryption
  - In open, synchronous systems $\Rightarrow$ Key agreements + digital signatures

Problem two

- Non-repudiation:
  - Anyone with the same pre-shared key can produce an authenticated message!
  - In open systems: digital signatures

Public Key Encryption



- $c \leftarrow_{\$} \mathrm{E}(pk, p)$ - Encryption with the public key
- $p \leftarrow \mathrm{D}(sk, c)$ - Decryption with the secret key

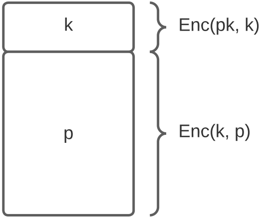
- **Intuition:**
  - Anyone with *pk* can encrypt
  - Only the one with *sk* can decrypt

Key Encapsulation Mechanisms

- Much more inefficient than symmetric ciphers
  - Asymmetric keys: thousands of bits (vs 128 bits)
  - Only feasible to encrypt very small messages
  - Payload: symmetric key

Key Encapsulation Mechanisms

- Much more inefficient than symmetric ciphers
  - Asymmetric keys: thousands of bits (vs 128 bits)
  - Only feasible to encrypt very small messages
  - Payload: symmetric key



- Hybrid paradigm (e.g. e-mail S/MIME)
  1. Sender generates symmetric session key k to encrypt p
  2. Sender gets pk, uses it to encrypt k
  3. Receiver gets two ciphertexts, corresponding to (1) and (2)

Building Public Key Encryption

Many options to build public key encryption!

Conceptually, quite simple

- Start with a mathematical object
- One-way trapdoor function
  - Given trapdoor x and s, I can find m s.t. $F(m) = s$

Building Public Key Encryption

Many options to build public key encryption!

Conceptually, quite simple

- Start with a mathematical object
- One-way trapdoor function
  - Given trapdoor x and s, I can find m s.t. $F(m) = s$
- One-way trapdoor permutation
  - Given trapdoor x and s, I can compute $F^{-1}(s)$
  - Most well-known trapdoor permutation: RSA

The RSA problem (Rivest, Shamir, Adleman 1977)

- Let e be a public exponent (typically `0x10001`)
- Select two large primes $p \cdot q$ (Nowadays, > 2048)
- Calculate $n \leftarrow p \cdot q$; $v \leftarrow (p-1) \cdot (q-1)$
- Compute d such that $d \cdot e \bmod v = 1$

The RSA problem (Rivest, Shamir, Adleman 1977)

- Let e be a public exponent (typically `0x10001`)
- Select two large primes $p \cdot q$ (Nowadays, > 2048)
- Calculate $n \leftarrow p \cdot q$; $v \leftarrow (p-1) \cdot (q-1)$
- Compute d such that $d \cdot e \bmod v = 1$

Hence

- $pk \leftarrow (e, n)$; $sk \leftarrow (d, n)$
- $F(pk, x) = x^e \bmod n$; $F^{-1}(sk, y) = y^d \bmod n$

$$x^{e \cdot d}\ mod\ N = x$$

OAEP Encryption

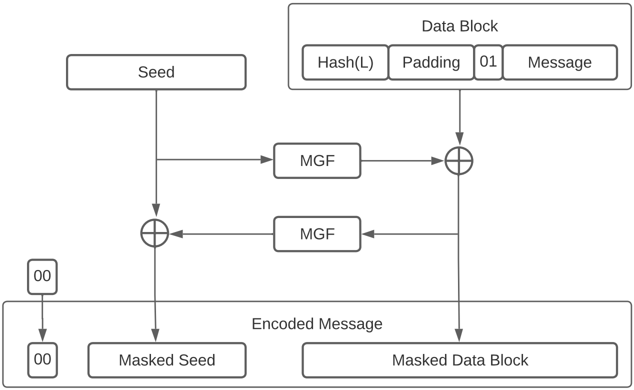
- Why can I not just use RSA to encrypt?
- For one, **it is not randomized!**

OAEP Encryption

- Why can I not just use RSA to encrypt?
- For one, **it is not randomized!**

OAEP construction

- Intelligently prepare the x that goes into RSA



Physical Signatures

Key properties

- Assurance of authorship of message/document
- Document not tampered after signature
  - Non-falsifiable
  - Non-reusable
  - Non-repudiation

Physical Signatures

Key properties

- Assurance of authorship of message/document
- Document not tampered after signature
  - Non-falsifiable
  - Non-reusable
  - Non-repudiation

Realistic Guarantees

- Many, many assumptions
- Think outside the box: can we subvert these guarantees?

Digital Signatures

Key properties

- Assurance of authorship of message/document
- Document not tampered after signature
  - Non-falsifiable
  - Non-reusable
  - Non-repudiation

Digital Signatures

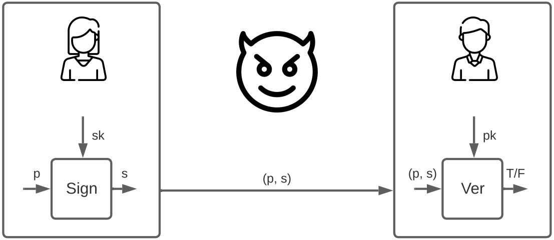
Key properties

- Assurance of authorship of message/document
- Document not tampered after signature
  - Non-falsifiable
  - Non-reusable
  - Non-repudiation

But they also entail assumptions!

- Signature key *sk* must not be compromised
- Algorithm cryptographically secure
- How can we enforce this?
  - PKI (soon)

Asymmetric Message Authentication



- $s \leftarrow_{\$} \text{Sign}(sk, p)$ - Encryption with the public key
- $T/\perp \leftarrow \text{Ver}(pk, p, s)$ - Decryption with the secret key
- **Intuition:**
    - Only the one with *sk* can sign
    - Anyone with *pk* can verify
        - Signature must go alongside the message
        - *s* can be orders of magnitude smaller than *p*

Assymetric Authenticity vs MACs

- Signatures ensure authenticity and integrity
- Similar to symmetric MACs!

Assymetric Authenticity vs MACs

- Signatures ensure authenticity and integrity
- Similar to symmetric MACs!

Context

- Does not require a pre-shared secret key
- Ensures **non-repudiation**

Non-repudiation

- Signee cannot deny the computation of the signature
- ... and thus its authorship!

Assymetric Authenticity vs MACs

- Signatures ensure authenticity and integrity
- Similar to symmetric MACs!

Context

- Does not require a pre-shared secret key
- Ensures **non-repudiation**

Non-repudiation

- Signee cannot deny the computation of the signature
- ... and thus its authorship!

Not the case for MACs. **Why?**

Example - Digital Signature with RSA

How can we build Digital signatures?

- Can also be done with RSA
  - $(pk, sk) \leftarrow Gen()$ - Just as in RSA
  - $\sigma \leftarrow F^{-1}(sk, H(m))$ - Secret exponent over the message digest
  - T iff $F(pk, H(m)) = \sigma$ - Invert signature, and check with message digest

Example - Digital Signature with RSA

How can we build Digital signatures?

- Can also be done with RSA
  - $(pk, sk) \leftarrow Gen()$ - Just as in RSA
  - $\sigma \leftarrow F^{-1}(sk, H(m))$ - Secret exponent over the message digest
  - T iff $F(pk, H(m)) = \sigma$ - Invert signature, and check with message digest

- Notice the hash over the message
- Otherwise anyone can forge the message 1, as $1^d = 1$
- Other issues also arise for related signatures

Application Context

Public Key Cryptography

- Authentication and non-repudiation with digital signatures
- Confidentiality with public key encryption
  - ... used to transport symmetric keys
- No requirement for *magical* key sharing
- **However**, we still need to assure their authenticity

Classical Use Case: Secure E-mail

Assumptions

- Alice knows Bob's public key $\rightarrow$ encrypt (pk_B)
- Bob knows Alices's public key $\rightarrow$ verify (pk_A)

Classical Use Case: Secure E-mail

Assumptions

- Alice knows Bob's public key $\rightarrow$ encrypt (pk_B)
- Bob knows Alices's public key $\rightarrow$ verify (pk_A)

Goals

- Message should be confidential
- Message should be authentic
- Non-repudiation of Alice's authorship

Non-Goals

- Alice knows that Bob received the message
- Alice knows that Bob accepted the message

Secure E-mail - Solution

Sign-then-Encrypt

- $\sigma \leftarrow \mathrm{Sign}(sk_A, m)$
- $c \leftarrow_{\$} \mathrm{Enc}(pk_B, m)$
- σ and c sent over the network

Secure E-mail - Solution

Sign-then-Encrypt

- $\sigma \leftarrow \mathsf{Sign}(sk_A, m)$
- $c \leftarrow_{\$} \mathsf{Enc}(pk_B, m)$
- σ and c sent over the network

Checking the Requirements

- Confidentiality = yes!

Secure E-mail - Solution

Sign-then-Encrypt

- $\sigma \leftarrow \text{Sign}(sk_A, m)$
- $c \leftarrow_{\$} \text{Enc}(pk_B, m)$
- σ and c sent over the network

Checking the Requirements

- Confidentiality = yes!
- Authenticity/Non-R = message was signed by Alice...
- But it might have not been originally to Bob!
  - Anyone can do $\text{Enc}(pk_B, m)$
  - Adversary can take something signed to him
  - ... and forward to Bob.

Careful on meta-data! Include recipient on the signed message.

Key Takeaways

- Managing symmetric keys is very hard
  - One key for each possible participant pair
  - Some centralized solutions

Key Takeaways

- Managing symmetric keys is very hard
  - One key for each possible participant pair
  - Some centralized solutions
- Key encapsulation mechanisms
  - Generate a symmetric key
  - Encrypt the payload with said symmetric key
  - Encrypt the key with public-key crypto

Key Takeaways

- Managing symmetric keys is very hard
  - One key for each possible participant pair
  - Some centralized solutions
- Key encapsulation mechanisms
  - Generate a symmetric key
  - Encrypt the payload with said symmetric key
  - Encrypt the key with public-key crypto
- RSA built on top of a “hard” problem
  - Can’t be used directly (either to encrypt or sign!)
  - RSA-OAEP

Key Takeaways

- Managing symmetric keys is very hard
  - One key for each possible participant pair
  - Some centralized solutions
- Key encapsulation mechanisms
  - Generate a symmetric key
  - Encrypt the payload with said symmetric key
  - Encrypt the key with public-key crypto
- RSA built on top of a “hard” problem
  - Can’t be used directly (either to encrypt or sign!)
  - RSA-OAEP
- Digital Signatures
  - The public-key equivalent of MACs
  - With bonus (or not): non-repudiation

Key Takeaways

- Managing symmetric keys is very hard
  - One key for each possible participant pair
  - Some centralized solutions
- Key encapsulation mechanisms
  - Generate a symmetric key
  - Encrypt the payload with said symmetric key
  - Encrypt the key with public-key crypto
- RSA built on top of a “hard” problem
  - Can’t be used directly (either to encrypt or sign!)
  - RSA-OAEP
- Digital Signatures
  - The public-key equivalent of MACs
  - With bonus (or not): non-repudiation
- Careful when combining signatures and encryptions!

Key Agreement Setting

Assumptions

- Alice knows Bob's public key (pk_B) to sign and verify
- Bob knows Alice's public key (pk_A) to sign and verify

Key Agreement Setting

Assumptions

- Alice knows Bob's public key (pk_B) to sign and verify
- Bob knows Alice's public key (pk_A) to sign and verify

Goals

- Establish a confidential session key (symmetric)
- Established key must be authentic and confirmed
- Perfect forward secrecy: compromise long-term keys should not compromise session keys

Non-Goals

- Non-repudiation

Key Agreement Applications

- Crucial for applications such as HTTPS
- Decades until we could converge to a secure and efficient solution
- Pks not used to transport symmetric keys
  - Otherwise no forward secrecy!
- Is based on the first paper on public key crypto: the **Diffie-Hellman** protocol
- Authentication $\rightarrow$ Digital signatures

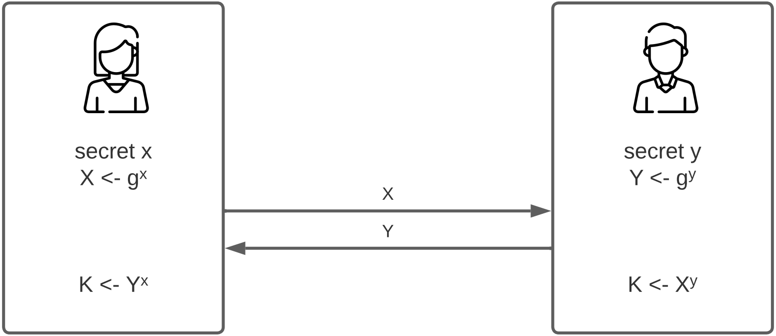
The Diffie-Hellman Protocol

Public parameters: finite group $(G, g, \cdot)$

- Group G
  - Values $[1..p[$ for some large prime number p
  - Points in an elliptic curve $\rightarrow$ used nowadays for efficiency
- Operation $\cdot$
  - Maps two group elements in a third
  - Commutative, associative, neutral element, etc.
- Group generator g
  - Allows us to encode a large integer to the group, irreversibly
  - For $e \in 0 \dots p-1$, $g^e = g \cdot g \cdot g(...) \cdot g$ produces different elements in G

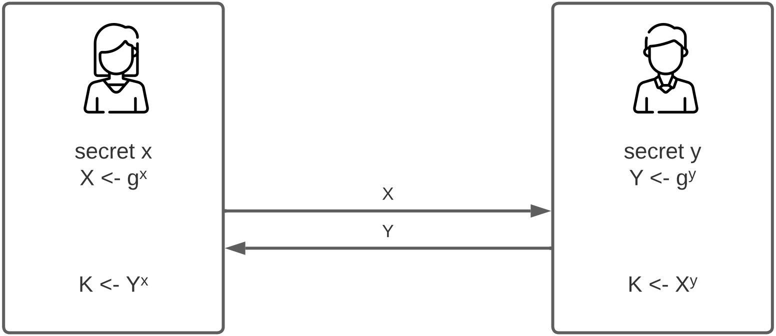
As such: $(g^x)^y = g^{xy} = g^{yx} = (g^y)^x$

DH in its Basic Form



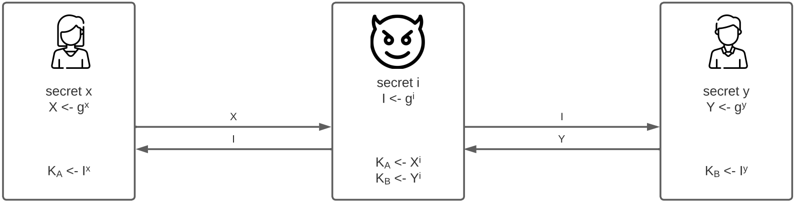
$$K = (g^y)^x = g^{yx} = g^{xy} = (g^x)^y$$

DH - Anonymity



- Observe that Alice might not be sure who Bob is
- ... and vice-versa
- This is intentional, and useful in many applications

Man-in-the-Middle



- K_A used to communicate with Alice
- K_B used to communicate with Bob
- If messages are simply forwarded, it's **undetectable**

Man-in-the-Middle in the Real-World

Attacks possible whenever:

- Public parameters are exchanged via the network
- Of potentially unknown origin.

Man-in-the-Middle in the Real-World

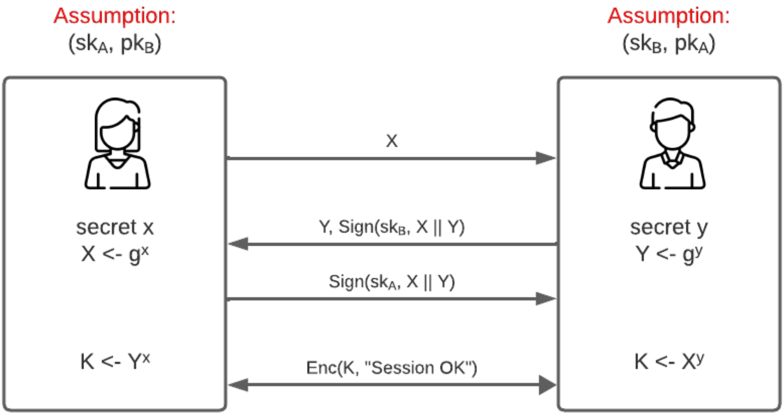
Attacks possible whenever:

- Public parameters are exchanged via the network
- Of potentially unknown origin.

Can happen in many contexts

- Public keys for signature/encryption
  - Everything we saw on RSA is subject to this!
- Public Diffie-Hellman parameters
- Any message, really...

Authenticated Diffie-Hellman



Establishing Secure Channels

Establishing secure channels entails

- Authenticated public keys $\rightarrow$ protect key agreement
- Key agreement protocols
  - Protects authenticity of symmetric keys
  - Ensures secrecy of keys for message confidentiality
  - Even if signature keys are corrupted in the future!
- Then, we can just use AEAD to exchange messages

Instantiate Secure Channels

- Designing a complete protocol is extremely complex
  - Even if individual components are individually secure
  - Their combination might not be
- TLS 1.3 (soon) is the result of 30 years of evolution
  - And there are still many issues to be resolved!
  - How to ensure a reasonable level of anonymity?

Next Challenge: Authentic Public Keys

Problem

- A key component for Alice to send an authentic key to Bob relies on reliable knowledge of public keys
- How can these be exchanged without interference?
- More formally: given pk_A, how can Bob knows this corresponds to the secret key sk_A, known only by Alice?

Next Challenge: Authentic Public Keys

Problem

- A key component for Alice to send an authentic key to Bob relies on reliable knowledge of public keys
- How can these be exchanged without interference?
- More formally: given pk_A, how can Bob knows this corresponds to the secret key sk_A, known only by Alice?

Solution

- Public-Key Infrastructure
- Trust in keys is validated by trust in a central authority
- Hierarchical Validation of Keys ensures scalability
- Not fool-proof, but pretty good

An Imminent Threat

From [1]

Quantum Computers change how we do computation

- q-bits allow the superposition of information
- Algorithms run over these q-bits for exponential speedup
- ... on specific problems!

[1] https://tinyurl.com/3as85wjk

For Crypto

Grover's Algorithm

- Quadratic speedup on testing keys
- For symmetric crypto, $2^n \rightarrow 2^{\frac{n}{2}}$
- AES keys have to double size
- 256 bits to 512. Not too big of a deal

For Crypto

Grover's Algorithm

- Quadratic speedup on testing keys
- For symmetric crypto, $2^n \rightarrow 2^{\frac{n}{2}}$
- AES keys have to double size
- 256 bits to 512. Not too big of a deal

Shor's Algorithm

- Find the prime factors of an integer "efficiently"
- For public-key crypto, BIG problem!
- We use RSA becase we assume that, for large primes p, q and $p * q = n$, one cannot get to p and q when given n!!
- Security of PK algorithms *reduces* to integer factorization

NIST call for PQC algorithms

A new flavor of challenge

- Lattice-based Cryptography
  - Learning With Errors (LWE)
  - Shortest Vector Problem (SVP)
  - Closest Vector Problem (CVP)
  - ...

NIST call for PQC algorithms

A new flavor of challenge

- Lattice-based Cryptography
  - Learning With Errors (LWE)
  - Shortest Vector Problem (SVP)
  - Closest Vector Problem (CVP)
  - ...

Replacements selected via public contest

- CRYSTALS-Kyber (Key Encapsulation Mechanism)
- NTRUEncrypt (Key Encapsulation Mechanism)
- CRYSTALS-Dilithium (Signatures)
- FALCON (Signatures)

From **Skepticism** to Paranoia

Should I care **now**?

From **Skepticism** to Paranoia

Should I care **now**? Short answer: Yes, definitely

- Store-now-decrypt-later (SNDL)
- or Harvest-now-decrypt-later

From **Skepticism** to Paranoia

Should I care **now**? Short answer: Yes, definitely

- Store-now-decrypt-later (SNDL)
- or Harvest-now-decrypt-later
- Obtain network communications today
- Store them for decades, if necessary
- When a quantum computer exists, use Shor's algorithm to extract private key, and thus decrypt the communication

From **Skepticism** to Paranoia

Should I care **now**? Short answer: Yes, definitely

- Store-now-decrypt-later (SNDL)
- or Harvest-now-decrypt-later
- Obtain network communications today
- Store them for decades, if necessary
- When a quantum computer exists, use Shor's algorithm to extract private key, and thus decrypt the communication

We currently have sufficient storage space to perform massive SNDL attacks, and have strong reasons to suspect these are currently well underway: link.

From Skepticism to **Paranoia**

On the other hand...

- Not reasonable to suspect quantum will break crypto
- At time of writing, Google's 70-qubit device, far below the 20 million necessary to break 1024-bit RSA
- Groundbreaking for very specialized tasks, but that's it

From Skepticism to **Paranoia**

On the other hand...

- Not reasonable to suspect quantum will break crypto
- At time of writing, Google's 70-qubit device, far below the 20 million necessary to break 1024-bit RSA
- Groundbreaking for very specialized tasks, but that's it

For the near future

- New algorithms lack maturity
- Improved network protocols (next class)
- Redundant usage of RSA and lattice-based crypto
- Double the toll, hopefully same security!

Key Takeaways

- Key agreement allows for symmetric crypto
- In public-key settings
  - Diffie-hellman allows for k to be exchanged
  - Provided one cannot solve the discrete logarithm

Key Takeaways

- Key agreement allows for symmetric crypto
- In public-key settings
  - Diffie-hellman allows for k to be exchanged
  - Provided one cannot solve the discrete logarithm
- Man-in-the-Middle
  - DH by itself lacks authentication
  - An interloper can just pose as participants
  - We need some way to ensure legitimacy of public keys
  - PKIs! (Next class)

Key Takeaways

- Key agreement allows for symmetric crypto
- In public-key settings
  - Diffie-hellman allows for k to be exchanged
  - Provided one cannot solve the discrete logarithm
- Man-in-the-Middle
  - DH by itself lacks authentication
  - An interloper can just pose as participants
  - We need some way to ensure legitimacy of public keys
  - PKIs! (Next class)
- Post-quantum Crypto
  - All modern public-key cryptography relies on problems solvable by quantum computers
  - Solutions are being presented
  - Still lack maturity
  - For now, we transition ASAP
  - And rely on a combination of classical and post-quantum

Computer Security Foundations
Week 10: Public Key Cryptography

Bernardo Portela

L.EIC - 24