OSY.SSI[2018][12]

- Iran in the midst of a near-total national internet shutdown amid protests [src].
- Russia new law to disconnect its internet from the rest of the world [src]
- Interpol plans to condemn encryption [src]

In the previous episode...

In the previous episode...

$$
\begin{aligned}
\mathcal{L} = &-\frac{1}{4} W_a^{\mu\nu} W^a_{\mu\nu} - \frac{1}{4} B^{\mu\nu} B_{\mu\nu} \\
&+ \overline{Q}_i i \not{D} Q_i + \overline{u}^c_i i \not{D} u^c_i + \overline{d}^c_i i \not{D} d^c_i + \overline{L}_i i \not{D} L_i + \overline{e}^c_i i \not{D} e^c_i \\
&+ |D_\mu h|^2 - \lambda \left(|h|^2 - \frac{v^2}{2} \right)^2 \\
&- y_{u\, ij} \epsilon^{ab} h^\dagger_b \overline{Q}_{ia} u^c_j - y_{d\, ij} h \overline{Q}_i d^c_j - y_{e\, ij} h \overline{L}_i e^c_j + hc \\
&+ \bar{q} \left(i \gamma^\mu D_\mu - m \right) q - \frac{1}{4} F^a_{\mu\nu} F_a^{\mu\nu}
\end{aligned}
$$

Part III
Security in the Real World^{TM}

Target, 2013

? September 2013	Attackers send a phishing e-mail to Fazio Mechanical Services, installing the Citadel malware and stealing VPN passwords.
15 November 2013	Attackers test BlackPoS malware on PoS machines.
27 November 2013	Attackers begin collecting credit card data.
30 November 2013	Attackers deploy BlackPoS on all machines, and install data exfiltration software.
30 November 2013	Security warnings raised by multiple security tools (FireEye, Symantec, etc.) but ignored by Target.
Up to December 2013	Attackers encrypt and store credit card numbers to three compromised internal FTP servers, then exfiltrate to servers in Miami, Brazil, and Russia.
2 December 2013	Additional security alerts are raised and ignored.
12 December 2013	US Department of Justice warned the company about suspicious activity involving payment cards, suggesting a potential breach. Target: "The team determined that it did not warrant immediate follow up."
15 December 2013	Target finds the BlackPoS software installed on its payment terminals.
18 December 2013	Attackers leak about 40 million accounts on the Internet. Media report the leak.
19 December 2013	Target publicly confirmed that some 40 million credit and debit card accounts were exposed in a breach of its network.
10 January 2014	Target confirms that in excess of 110 million accounts were in fact stolen.
6 February 2014	Fazio Mechanical Services official statement, its "data connection with Target was exclusively for electronic billing, contract submission and project management." and its "IT system and security measures are in full compliance with industry practices."
March 2014	Target: "With the benefit of hindsight, we are investigating whether if different judgments had been made the outcome may have been different."
April 2014	Target CEO Gregg Steinhafel resigns.

Addendum: $18.5 Million Breach Settlement with states

Timeline of the 2011 RSA SecurID incident.

28 February 2011	`@yuange1975` posts a proof-of-concept heap overflow for Adobe Flash on Twitter.
1 March 2011	Attackers send a first e-mail with a crafted Adobe Flash object included in an Excel file.
2 March 2011	Attackers send a second e-mail. User opens the attached Excel file, triggering the exploit.
10 March 2011	RSA discovers the attack, reports to executives.
14 March 2011	Adobe Security Advisory about the Flash vulnerability, given reference CVE-2011-0609.
17 March 2011	RSA announces to customers they had been victims of "an extremely sophisticated cyber attack".
17 March 2011	RSA writes a formal Form 8-K, indicating the breach was unlikely to have a "material impact on its financial results".
21 March 2011	Adobe releases patch for CVE-2011-0609.
27 May 2011	Lockheed Martin systems attacked, attackers leveraged RSA compromise.
31 May 2011	L-3 Communications attacked, attackers leveraged RSA compromise.
1 June 2011	Northrop Grumman attacked, attackers leveraged RSA compromise.
6 June 2011	RSA admits SecureID compromise, replaces millions of tokens. Company chairman says "we believe and still believe that the customers are protected".
1 August 2011	Financial impact of the breach on RSA estimated at about $66.3 million.
22 August 2011	The initial phishing e-mail is found by investigators.

Timeline of the 2015 TV5Monde incident.

23 January 2015	Attackers scan and find a default password on a RDP server. B ut this is a dead-end.
6 February 2015	Attackers access the internal network through stolen VPN credentials. They scan the network and find two Windows machines controlling cameras.
11 February 2015	Attackers use one of these machines to create a new Active Directory administrator account.
16-25 February	Attackers collect data (files, internal wiki, etc.) and login and password information. They verify that the passwords still work. Attackers compromise another administrator machine and install the Sofacy RAT.
3 April 2015	Attackers get access to TV5Monde's social networks accounts and tests the credentials, but do not modify anything.
8 April 19:57	Attackers reconfigure IPs for the encoders. This is not noticed, as the misconfiguration only gets enabled when the technical teams reboot the machines.
8 April 20:58	Attackers posted messages on the channel's Twitter, YouTube, and Facebook pages. They also replace TV5Monde's website by a pro-ISIS page.
8 April 21:48	Attacker erase the firmwares from the switches and routers, effectively stopping 12 channels from airing.
8 April 22:40	Attackers delete the internal e-mail server.
8 April 2015	Around midnight, a decision is made to "cut the cord" to prevent attackers from causing further damage. ANSSI experts arrive the next morning.
9 April 20:00	Pre-recorded messages can be aired by TV5Monde.
11 May 2015	A clean system is ready and content is being migrated, restoring full functionality.

What do they have in common?

- These three attacks may seem quite different: different motives, different companies, different impact, etc.

What do they have in common?

- These three attacks may seem quite different: different motives, different companies, different impact, etc.
- But from a strategic point of view, the attackers followed a similar approach:

What do they have in common?

- These three attacks may seem quite different: different motives, different companies, different impact, etc.
- But from a strategic point of view, the attackers followed a similar approach:
  - Reconnaissance.
    They knew the organisation very well, and did not wait for an opportunity to start planning. They had an idea of the technologies used and investigated employees.
  - Entry point.
    Using their knowledge, they find a way to enter the organisation's perimeter.
  - Lateral (or horizontal) evolution. Exploring around their entry point, they manage to hop from one device to another in the same area/security level. They would install a RAT to facilitate this step, and collect information.
  - Vertical evolution (or privilege escalation).
    This new information allows them to attempt getting access to a more secure (or more interesting) position.
  - Evasion.
    Either by disclosing the vulnerability (RSA), leaking some data (Target), or using false flag techniques (TV5), attackers distract from the real nature and goal of their operation.
  - Monetisation.
    Having succeeded, attackers reap the benefits of their deeds, by reselling (Target) or reusing (RSA) the bounty. TV5 is particular as it seems the attackers just wanted to destroy it.

What do they have in common?

We therefore suggest to use the following interpretative guidelines:

1. Recon
2. Intel/Vuln discovery/Weaponisation
3. Entry/Turning/Pivoting
4. Evolution (H/V)
5. Payload delivery/Exfiltration
6. Evasion/Diversion
7. Monetisation/Mediatisation

This is known as The Killchain.

Reminder: Defence in Depth

1. Prevent intrusion by setting up boundaries
2. Resist crossing to delay the adversary and force them inside one area
3. Limit the impact of area intrusion (PLP)
4. Detect area intrusion (how?)
5. Log area intrusions

Reminder: Defence in Depth

1. Prevent intrusion by setting up boundaries
2. Resist crossing to delay the adversary and force them inside one area
3. Limit the impact of area intrusion (PLP)
4. Detect area intrusion (how?)
5. Log area intrusions

Is that enough?

Reminder: Defence in Depth

1. Prevent intrusion by setting up boundaries
2. Resist crossing to delay the adversary and force them inside one area
3. Limit the impact of area intrusion (PLP)
4. Detect area intrusion (how?)
5. Log area intrusions

Is that enough? Audit!... and test!

Elements of counter-strategy

Log, Detect, Limit, Prevent

Elements of counter-strategy

Log, Detect, Limit, Prevent

- For every step of the adversarial strategy

Elements of counter-strategy

Log, Detect, Limit, Prevent

- For every step of the adversarial strategy
- How? Three (complementary) approaches:

Elements of counter-strategy

Log, Detect, Limit, Prevent

- For every step of the adversarial strategy
- How? Three (complementary) approaches:
  - Top-down: Architecture / Resilience and response

Elements of counter-strategy

Log, Detect, Limit, Prevent

- For every step of the adversarial strategy
- How? Three (complementary) approaches:
  - Top-down: Architecture / Resilience and response
  - Bottom-up: Tools and crypto / Security by design

Elements of counter-strategy

Log, Detect, Limit, Prevent

- For every step of the adversarial strategy
- How? Three (complementary) approaches:
  - Top-down: Architecture / Resilience and response
  - Bottom-up: Tools and crypto / Security by design
  - Empirical: Audit and pentest / Continuous improvement

Elements of counter-strategy

Log, Detect, Limit, Prevent

- For every step of the adversarial strategy
- How? Three (complementary) approaches:
  - Top-down: Architecture / Resilience and response
  - Bottom-up: Tools and crypto / Security by design
  - Empirical: Audit and pentest / Continuous improvement

Counter-strategic or “Depth” boundaries

1. Recon

Counter-strategic or “Depth” boundaries

1. Recon $\rightarrow$ Information policy / Change
2. Intel/Vuln discovery/Weaponisation

Counter-strategic or “Depth” boundaries

1. Recon → Information policy / Change
2. Intel/Vuln discovery/Weaponisation → Honeypot / Change / Intox
3. Entry/Turning/Pivoting

Counter-strategic or "Depth" boundaries

1. Recon → Information policy / Change
2. Intel/Vuln discovery/Weaponisation → Honeypot / Change / Intox
3. Entry/Turning/Pivoting → Alerting / PLP / Small areas
4. Evolution (H/V)

Counter-strategic or “Depth” boundaries

1. Recon → Information policy / Change
2. Intel/Vuln discovery/Weaponisation → Honeypot / Change / Intox
3. Entry/Turning/Pivoting → Alerting / PLP / Small areas
4. Evolution (H/V) → Alerting / PLP / Small areas
5. Payload delivery/Exfiltration

Counter-strategic or "Depth" boundaries

1. Recon → Information policy / Change
2. Intel/Vuln discovery/Weaponisation → Honeypot / Change / Intox
3. Entry/Turning/Pivoting → Alerting / PLP / Small areas
4. Evolution (H/V) → Alerting / PLP / Small areas
5. Payload delivery/Exfiltration → PLP / Small areas
6. Evasion/Diversion

Counter-strategic or “Depth” boundaries

1. Recon → Information policy / Change
2. Intel/Vuln discovery/Weaponisation → Honeypot / Change / Intox
3. Entry/Turning/Pivoting → Alerting / PLP / Small areas
4. Evolution (H/V) → Alerting / PLP / Small areas
5. Payload delivery/Exfiltration → PLP / Small areas
6. Evasion/Diversion → Auditing / Event correlation
7. Monetisation/Mediatisation

Counter-strategic or "Depth" boundaries

1. Recon → Information policy / Change
2. Intel/Vuln discovery/Weaponisation → Honeypot / Change / Intox
3. Entry/Turning/Pivoting → Alerting / PLP / Small areas
4. Evolution (H/V) → Alerting / PLP / Small areas
5. Payload delivery/Exfiltration → PLP / Small areas
6. Evasion/Diversion → Auditing / Event correlation
7. Monetisation/Mediatisation → Change

Temporary conclusion

- Trust, but verify
- Layered defence mechanism to enable action and gather information
- Align defence strategy against adversarial strategy
- Plan for the worst-case scenario, and build upwards
- Freedom of movement is often preferable to firepower
- Beware of all the bullshit you can find on the Internet (or worse: elsewhere)

Security “by design”

Idea:

- Formally (=mathematically) define a security goal
- Design a system that satisfies this property (under reasonable hypotheses)
- Try to implement your system in the real world

In other terms, we want a provable guarantee that the security property is satisfied.

The prime example is *cryptography*, but let's get there slowly.

Early attempts: Steganography

For a long time, stenanography was used.

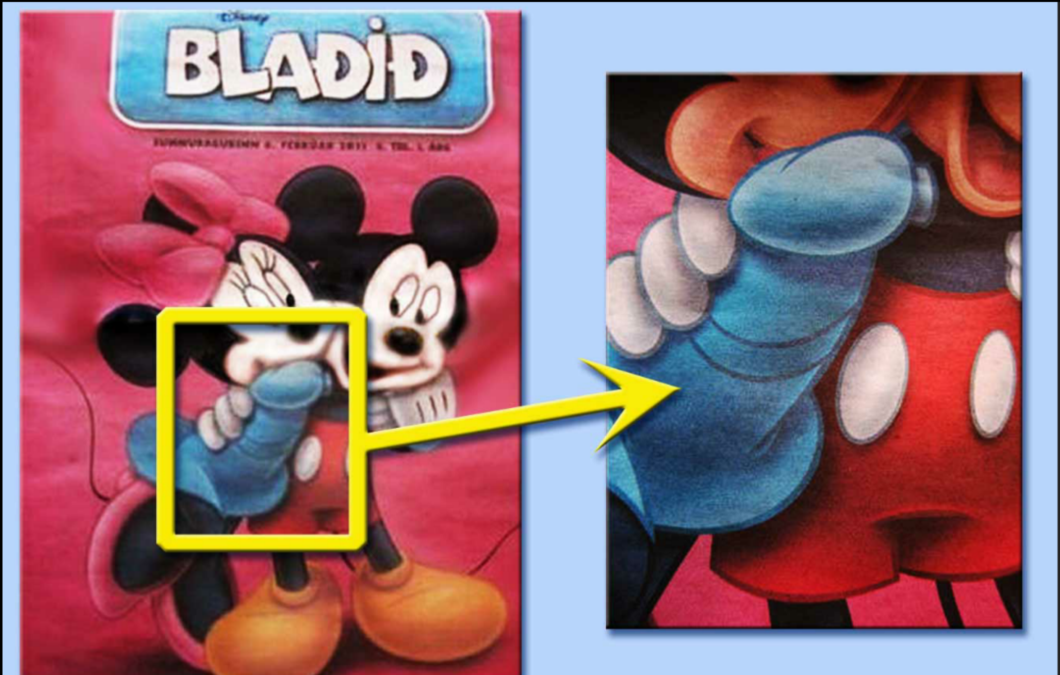
Steganography

Hiding a message (in plain sight).

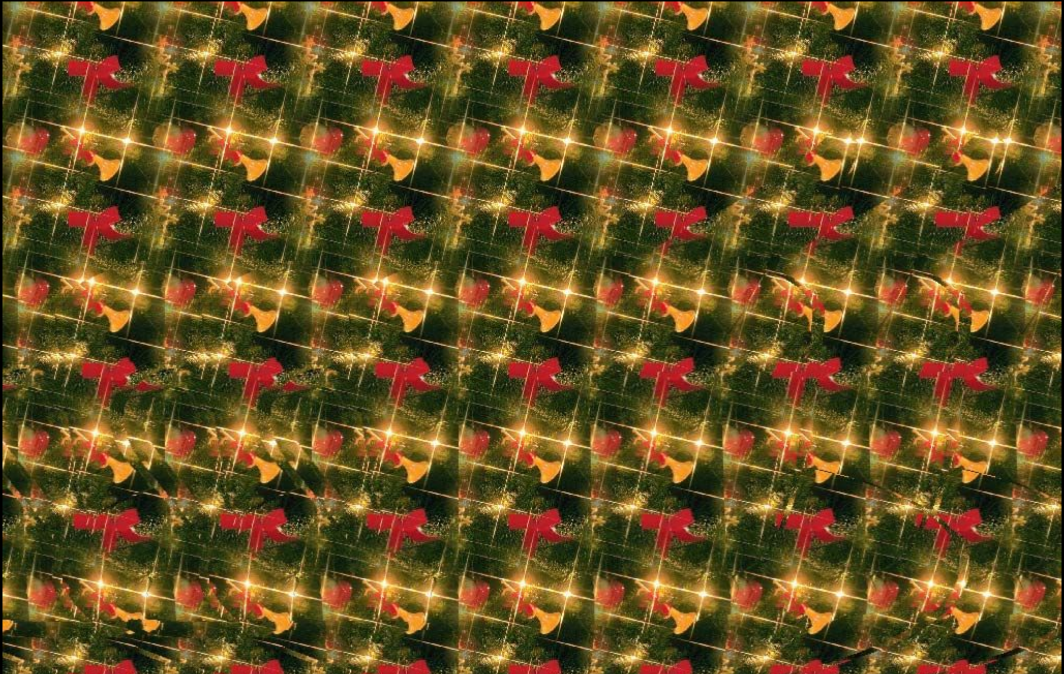
Steganography: Examples

Steganography: Examples

Steganography: Examples



Steganography: Examples



What's wrong with steganography?

What's wrong with steganography?

The message is there.

If the adversary knows or finds where to look, game's over.

No guarantee.

Defining our security objectives

Defining our security objectives

- The adversary shouldn't learn anything from the message.
  The message should "look" random (Indistinguishability)

Defining our security objectives

- The adversary shouldn't learn anything from the message.
  The message should "look" random (Indistinguishability)
- The adversary should not be helped by knowing the system The system is public (Kerckhoffs' principle)

Bonus: It should be easy to use the system in real life The system is algorithmically tractable (Polynomial-time)

Let's see the effect of these constraints

Kerckhoffs' principle

Confidentiality doesn't come from the (public) system.
A special input, called the key, is kept private.
The secrecy of this key should guarantee confidentiality.

Ideally:

Knowing the key $\Leftrightarrow$ Reading the message

Indistinguishability

Ideally:

Each encryption is different, and the distance

$$\Delta(M, X) = \frac{1}{2} \sum_{\omega \in \Omega} |\Pr(M = \omega) - \Pr(X = \omega)|$$

is negligible when X is uniformly distributed.

Example: One time pad

Message: $m \in \{0,1\}^n$.
Key: $k \in \{0,1\}^n$ uniformly at random.

$$\mathrm{Enc}(m,k) = m \oplus k \tag{1}$$
$$\mathrm{Dec}(c,k) = c \oplus k \tag{2}$$

Theorem (Mauborgne, Shannon)

OTP encryption is secure.

Example: One time pad... ?

1. Can we generate random numbers with a computer?
2. The key is as long as the message! And more secret!
3. Can we reuse the key?
4. Can we communicate with a new person?

Table of Contents

Statistics vs. complexity

OTP is the only known encryption scheme that is secure against all adversaries. But it is very, very impractical.

Instead of an “impossible to break” system, we will focus on those that are “hard to break”.

What do we mean, “hard”?

Algorithmic complexity

Algorithmic time complexity

An algorithm is said to have time complexity $O(f(n))$ if, given an input of size n, this algorithm terminates in time $O(f(n))$.

Example: Sort is $O(n \log n)$.

Example: **P** and **NP**

P : Problems for which we can find a solution in time $O(\text{poly})$.
NP : Problems for which we can check a solution in time $O(\text{poly})$

In particular, $\mathbf{P} \subset \mathbf{NP}$.

Exercice: Prove that $\mathbf{P} = \mathbf{NP}$.

P and NP problems

Problems in **P**:

- linear programming, greatest common divisor
- Type inference
- Determining if a number is prime

Problems in **NP**:

- Hamiltonian path
- Traveling salesman problem
- Knapsack / Subset-sum

General intuition: **P** is “easy”, **NP** is “interesting”

Hard problems in crypto

We focus on the following two problems:

- Integer factorisation: Given $n \in \mathbb{N}$, find d such that $d|n$.
- Discrete logarithm: Given $h \in G = \langle g \rangle$, find x such that $h = g^x$.

There are many others, but these are simple, well-known, and widespread.

There problems are believed to be hard to solve in practice for well-chosen n or G.

Table of Contents

Textbook RSA

A very early encryption scheme (1976) is RSA.

Setup : Choose p, q primes and set $n = pq$. Find d such that $3d \equiv 1 \bmod (p-1)(q-1)$.

Encryption : Choose $m \in \{0, \ldots, n-1\}$. Let $c = m^3 \bmod n$.

Decryption : Recover $m = c^d \bmod n$

It works because $(m^3)^d \equiv m^{3d} \equiv m^1$.

d constitutes the private key that allows decryption.
n is the public key that allows encryption.

Does it satisfy our security goals?

- Kerckhoffs' principle? Yes!
- Algorithmically tractable? Yes!
- Indistinguishability?

Does it satisfy our security goals?

- Kerckhoffs' principle? Yes!
- Algorithmically tractable? Yes!
- Indistinguishability? No!

If encryption is deterministic, it cannot be semantically secure.

Also what is the encryption of $m_1 \times m_2$?

Does it satisfy our security goals?

- Kerckhoffs' principle? Yes!
- Algorithmically tractable? Yes!
- Indistinguishability? No!

If encryption is deterministic, it cannot be semantically secure.

Also what is the encryption of $m_1 \times m_2$?

This can be fixed by using a padding function.

How secure is (padded) RSA?

- Integer factorisation breaks RSA (if I can factor n, then I can compute d.
- Integer factorisation is sufficient, but not necessary to break RSA.
- That being said, it is the best known way to attack in general

Best factorisation algorithm: Number field sieve, heuristic complexity

$$\exp\left(\left(\sqrt[3]{\frac{64}{9}}+o(1)\right)(\ln n)^{\frac{1}{3}}(\ln\ln n)^{\frac{2}{3}}\right)$$

current record RSA-768 (768 bits).

For a security level of 2^{128} (civilian use) we get $n \approx 2^{3072}$.

RSA: Key generation is essential!

Some keys are more vulnerable than others...
Example: Assume $n = pq$ and $n' = pq'$. Then $\gcd(n, n') = p$ and you can factor both n and n'.
Lenstra et al. in 2012:

1. Crawl the internet for public keys
2. Compute pairwise GCD (fast!)
3. ???
4. Profit!

Results: $> 1.4\%$ of RSA keys broken.

Note: RSA is mainly used as a signature today (with the PSS padding), although it is slowly being phased out and remplaced by DLP-based alternatives.

DLP-based alternatives

Cyclic group

A finite group G completely generated by an element g is cyclic :

$$G = \langle g \rangle = \{1, g, g^2, \ldots, g^{q-1}\}$$

g is called a “generator” of G.
The order of G is the smallest positive integer q s.t. $g^q = 1$.

Example: $G = \mathbb{F}_7^\times$ is generated by $g = 3$: $\langle g \rangle = \{1, 3, 2, 6, 4, 5\}$

Reminder: DLP

Discrete logarithm problem

Given $h \in \langle g \rangle$, find a such that $h = g^a$.

In a generic group of prime order p, the best-known algorithm is in $O(\sqrt{p})$ evaluations.

$\mathbb{F}_p$ is *not* a generic group, faster algorithms exist (index calculus). We'll come back to that later.

DDH problem

Decisional Diffie-Hellman problem

Given (g^a, g^b, g^c) tell whether $c = ab$ or not, better than chance.

Computational Diffie-Hellman problem

Given (g^a, g^b), output g^{ab}.

Break DLP $\Rightarrow$ break CDH, and break CDH $\Rightarrow$ break DDH.

Diffie-Hellman key exchange (DHE)

Alice and Bob agree on $\langle g \rangle$ (public).

Alice:

1. Choose a at random
2. Send $A = g^a$ to Bob.

Bob:

1. Choose b at random
2. Send $B = g^b$ to Alice.

Now Alice computes $B^a = g^{ab}$ and Bob computes $A^b = g^{ab}$.
They have the same result!

MITM

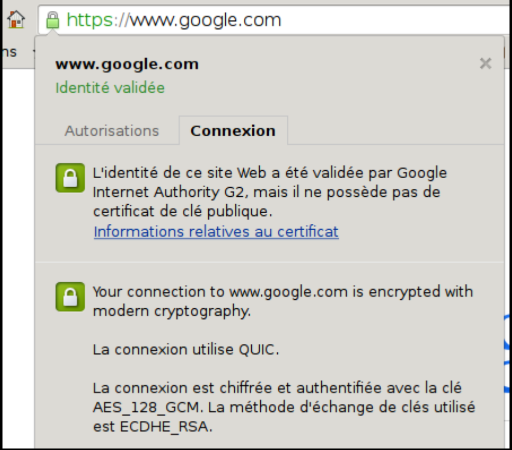
An adversary can intercept messages between Alice and Bob.

By exchanging keys with Bob on one hand, and Alice on the other, the adversary can impersonate both.

How to prevent this from happening?
Authenticated DHE, with e.g. RSA or DSA signature.

Beware of implementations (e.g. Logjam, Minerva, etc).

Note: This is ubiquitous



ECDHE_RSA stands for (Elliptic Curve) Diffie-Hellman key-exchange, authenticated with RSA signatures. What's AES?

Hybrid PK-SK

The public-key primitives that we have discussed (signature, encryption, key exchange) are mathematically backed, but terribly slow for most applications.

In practice, we negociate a large key K once per session, using authenticated DHE, and then turn to faster, but possibly weaker symmetric encryption methods (i.e. AES).

What do we do with cryptography?

From a ITsec perspective

What do we do with cryptography?

From a ITsec perspective

One main and most ubiquitous use of crypto is to provide **integrity** guarantees.

Example: Alice sends a message to Bob. How does he know the message is correct ?

What do we do with cryptography?

From a ITsec perspective

One main and most ubiquitous use of crypto is to provide **integrity** guarantees.

Example: Alice sends a message to Bob. How does he know the message is correct ?

- Error-correcting codes work only up to a point

What do we do with cryptography?

From a ITsec perspective

One main and most ubiquitous use of crypto is to provide **integrity** guarantees.

Example: Alice sends a message to Bob. How does he know the message is correct ?

- Error-correcting codes work only up to a point
- An adversary could intercept the message, and generate another one, with the correct ECCs

What do we do with cryptography?

From a ITsec perspective

One main and most ubiquitous use of crypto is to provide **integrity** guarantees.

Example: Alice sends a message to Bob. How does he know the message is correct ?

- Error-correcting codes work only up to a point
- An adversary could intercept the message, and generate another one, with the correct ECCs

Cryptography provides instead signatures and message authentication codes (MAC).

What do we do with cryptography?

From a ITsec perspective

One main and most ubiquitous use of crypto is to provide **integrity** guarantees.

Example: Alice sends a message to Bob. How does he know the message is correct ?

- Error-correcting codes work only up to a point
- An adversary could intercept the message, and generate another one, with the correct ECCs

Cryptography provides instead signatures and message authentication codes (MAC).

MAC Algorithms use a *secret key* and a message, they output a *digest*.

What do we do with cryptography?

From a ITsec perspective

One main and most ubiquitous use of crypto is to provide **integrity** guarantees.

Example: Alice sends a message to Bob. How does he know the message is correct ?

- Error-correcting codes work only up to a point
- An adversary could intercept the message, and generate another one, with the correct ECCs

Cryptography provides instead signatures and message authentication codes (MAC).

MAC Algorithms use a *secret key* and a message, they output a *digest*.

Question: assumptions? weaknesses? properties?

What do we do with cryptography?

From a ITsec perspective

One main and most ubiquitous use of crypto is to provide **integrity** guarantees.

Example: Alice sends a message to Bob. How does he know the message is correct ?

- Error-correcting codes work only up to a point
- An adversary could intercept the message, and generate another one, with the correct ECCs

Cryptography provides instead signatures and message authentication codes (MAC).

MAC Algorithms use a *secret key* and a message, they output a *digest*.

Question: assumptions? weaknesses? properties?
Question: encrypt and MAC? encrypt then MAC? MAC then encrypt?

What do we do with cryptography?

From a ITsec perspective

One main and most ubiquitous use of crypto is to provide **integrity** guarantees.

Example: Alice sends a message to Bob. How does he know the message is correct ?

- Error-correcting codes work only up to a point
- An adversary could intercept the message, and generate another one, with the correct ECCs

Cryptography provides instead signatures and message authentication codes (MAC).

MAC Algorithms use a *secret key* and a message, they output a *digest*.

Question: assumptions? weaknesses? properties?
Question: encrypt and MAC? encrypt then MAC? MAC then encrypt?
Encrypt-then-MAC.

What you'll need to understand

Even if you're not a cryptographer

1. How to *correctly use it* (what to encrypt, how, what mode, what blocksize?, where to sign, when to MAC? etc.)

What you'll need to understand

Even if you're not a cryptographer

1. How to *correctly use it* (what to encrypt, how, what mode, what blocksize?, where to sign, when to MAC? etc.)
2. How to manage keys? How to manage randomness?

What you'll need to understand

Even if you're not a cryptographer

1. How to *correctly use it* (what to encrypt, how, what mode, what blocksize?, where to sign, when to MAC? etc.)
2. How to manage keys? How to manage randomness?
3. How to *correctly implement cryptographic algorithms*

What you'll need to understand

Even if you're not a cryptographer

1. How to *correctly use it* (what to encrypt, how, what mode, what blocksize?, where to sign, when to MAC? etc.)
2. How to manage keys? How to manage randomness?
3. How to *correctly implement cryptographic algorithms*

Caveat – We're toying with live things

Either this is a pause, and more crypto follows.
Or this is the end of the lecture, and a surprise follows.