Computer Security

INFR10067

Fall 2025

Cryptography

Introduction

Markulf Kohlweiss
School of Informatics
University of Edinburgh

What is cryptography?

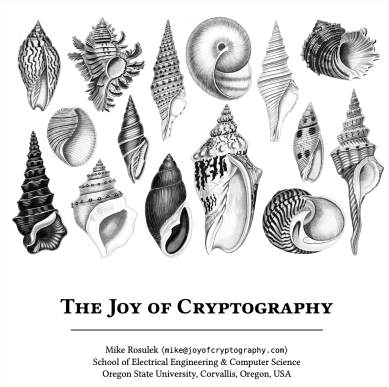
> *"The practice of creating and understanding codes that keep information secret."*
>
> Cambridge dictionary

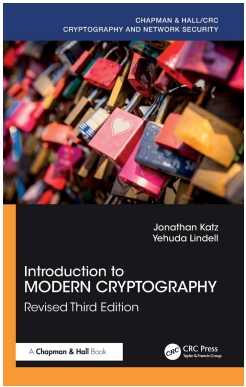
But nowadays cryptography encompasses many more things than just secret communications.

> *"Cryptography is the scientific study of techniques for securing [against internal or external attacks] digital information, transactions, and distributed computations."*
>
> Jonathan Katz and Yehuda Lindell
> in Introduction to Modern Cryptography

Acknowledgements and Textbooks

- Textbooks:





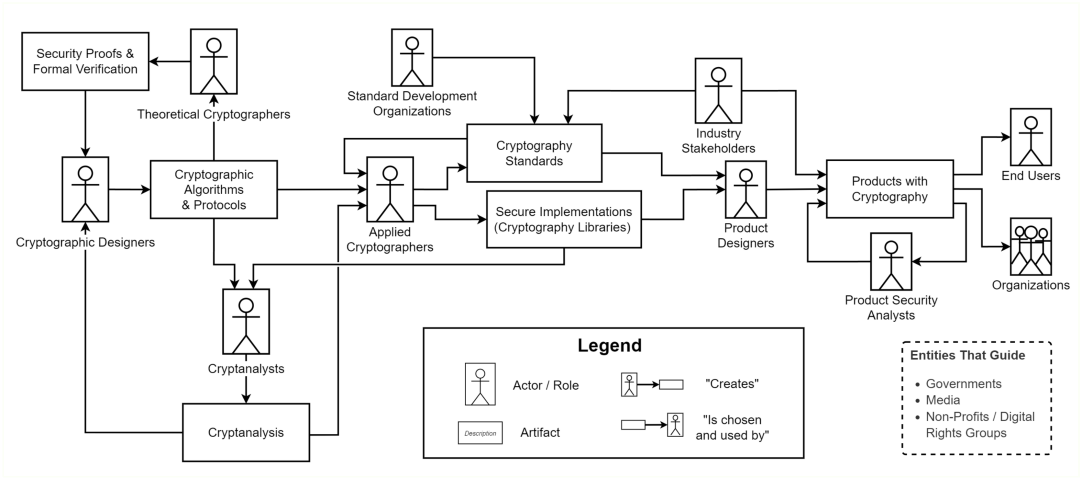
- Slides adapted from Myrto Arapinis and Nadim Kobeissi.
  Beamer style curtesy to Nadim Kobeissi: https://appliedcryptography.page

Cryptography is everywhere

- Banking
- Buying stuff from the store
- Any digital payment system
- Messaging (WhatsApp, Signal, iMessage, Telegram)
- Voice calls and video conferencing
- Government and military systems
- SSH
- VPN access
- Visiting most websites (HTTPS)

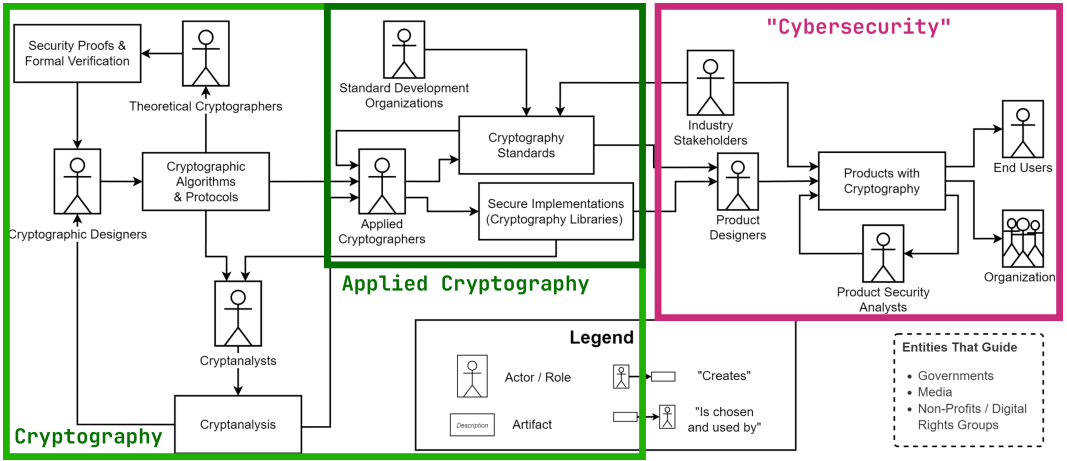
- Disk encryption
- Cloud storage
- Password managers
- Unlocking your car
- Identity card systems
- Ticketing systems
- DRM solutions
- Private contact discovery
- Cryptocurrencies

How it's made



Fischer et al., The Challenges of Bringing Cryptography from Research Papers to Products: Results from an Interview Study with Experts, USENIX Security 2024

How it's made



Fischer et al., The Challenges of Bringing Cryptography from Research Papers to Products: Results from an Interview Study with Experts, USENIX Security 2024

Important remark

Cryptography is not:

- The solution to all security problems (see other sections of the course)
- Secure if not implemented and/or deployed correctly
- Something you will be able to invent at the end of this course

Learning objectives for the Cryptography section

- Appreciate the variety of applications that use cryptography with different purposes
- Introduce the basic concepts of cryptography
- Understand the type of problems cryptography can address
- Understand the types of problems that need to be addressed when using cryptography

Topics in the Cryptography section

We will discuss constructions for:

- Symmetric Encryption
- Asymmetric (public-key) Encryption
- Hash functions and Message Authentication Codes (MACs)
- Digital Signatures and Public Key Infrastructure (PKI)
- Post Quantum Security

We present only the rudiments of the topic:

- What cryptography can achieve
- That cryptography can go wrong
- What is good practice when using cryptography

THE UNIVERSITY of EDINBURGH

Computer Security

INFR10067

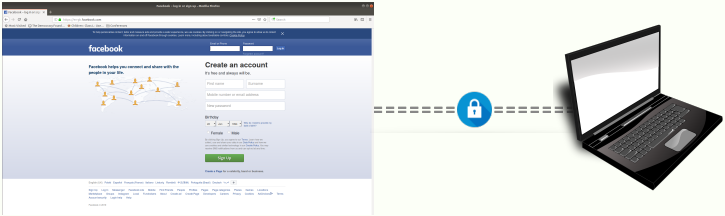
Fall 2025

Cryptography

Symmetric encryption

Goal: confidentiality

- Secure communications



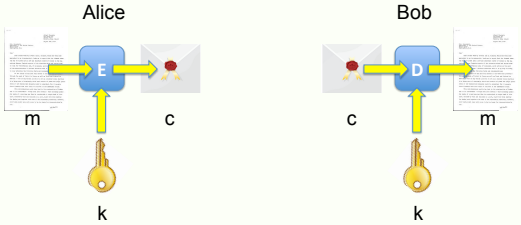
- File protection

Symmetric encryption schemes

A symmetric cipher consists of two algorithms

- encryption algorithm $E : \mathcal{K} \times \mathcal{M} \to \mathcal{C}$
- decryption algorithm $D : \mathcal{K} \times \mathcal{C} \to \mathcal{M}$

st. $\forall k \in \mathcal{K}$, and $\forall m \in \mathcal{M}, D(k, E(k, m)) = m$



- same key k to encrypt and decrypt
- the key k is secret: only known to Alice and Bob

What is a good encryption scheme?

An encryption scheme is secure against a given adversary, if this adversary cannot

- recover the secret key k
- recover the plaintext m underlying a ciphertext c
- recover any bits of the plaintext m underlying a ciphertext c
- ...

Kerckhoff's principle

The architecture and design of a security system/mechanism should be made public

No security through obscurity!

- The encryption (E) and decryption (D) algorithms are public
- The security relies entirely on the secrecy of the key

Open design allows for a system to be scrutinized by many users, white hat hackers, academics, *etc.*

⇝ early discovery and corrections of flaws/vulnerabilities

Adversary's capabilities

- A cryptographic scheme is secure under some assumptions, that is against a certain type of attacker
- A cryptographic scheme may be vulnerable to certain types of attacks but not others

The attacker know the encryption/decryption algorithms but may have access to :

- Ciphertext only attack - some ciphertexts c_1, ..., c_n
- Known plaintext attack some plaintext/ciphertext pairs (m_1, c_1), ..., (m_n, c_n) st. $c_i = E(k, m_i)$)
- Chosen plaintext attack - he has access to an encryption oracle - can maybe trick a user to encrypt messages m_1, ..., m_n of his choice
- Chosen ciphertext attack - he has access to a decryption oracle - can maybe trick a user to decrypt ciphertexts c_1, ..., c_n of his choice
- unlimited, or polynomial, or realistic ($\leq 2^{80}$) computational power

Brute-force attack - attack on all schemes

- Try all possible keys $k \in \mathcal{K}$ - requires some knowledge about the structure of plaintext

- Making exhaustive search unfeasible:
  - $\mathcal{K}$ should be sufficiently large, *i.e.* keys should be sufficiently long
  - Keys should be sampled uniformly at random from $\mathcal{K}$

A simple scheme: the substitution cipher

- shared secret: a permutation π of the set of characters

  $\pi = \; a \mapsto q \; b \mapsto w \; c \mapsto e \; d \mapsto r \; e \mapsto t \; f \mapsto y \; g \mapsto u \; h \mapsto i \; i \mapsto o$
  $j \mapsto m \; k \mapsto a \; l \mapsto s \; m \mapsto d \; n \mapsto f \; o \mapsto g \; p \mapsto h \; q \mapsto j \; r \mapsto k$
  $s \mapsto l \; t \mapsto z \; u \mapsto x \; v \mapsto c \; w \mapsto v \; x \mapsto b \; y \mapsto n \; z \mapsto p$

- Encryption: apply π to each character of the plaintext

$$E(\pi, p_1 \dots p_n) \; = \; \pi(p_1) \dots \pi(p_n)$$

- Decryption: apply π^{-1} to each character of the plaintext

$$D(\pi, c_1 \dots c_n) \; = \; \pi^{-1}(c_1) \dots \pi^{-1}(c_n)$$

Substitution cipher: example

$\pi =$ $a \mapsto q\ b \mapsto w\ c \mapsto e\ d \mapsto r\ e \mapsto t\ f \mapsto y\ g \mapsto u\ h \mapsto i\ i \mapsto o\ j \mapsto m\ k \mapsto a\ l \mapsto s$
$m \mapsto d\ n \mapsto f\ o \mapsto g\ p \mapsto h\ q \mapsto j\ r \mapsto k\ s \mapsto l\ t \mapsto z\ u \mapsto x\ v \mapsto c\ w \mapsto v\ x \mapsto b\ y \mapsto n\ z \mapsto p$

m = THIS COURSE AIMS TO INTRODUCE YOU TO THE PRINCIPLES AND TECHNIQUES OF SECURING COMPUTERS AND COMPUTER NETWORKS WITH FOCUS ON INTERNET SECURITY. THE COURSE IS EFFECTIVELY SPLIT INTO TWO PARTS. FIRST INTRODUCING THE THEORY OF CRYPTOGRAPHY INCLUDING HOW MANY CLASSICAL AND POPULAR ALGORITHMS WORK E.G. DES, RSA, DIGITAL SIGNATURES, AND SECOND PROVIDING DETAILS OF REAL INTERNET SECURITY PROTOCOLS, ALGORITHMS, AND THREATS, E.G. IPSEC, VIRUSES, FIREWALLS. HENCE, YOU WILL LEARN BOTH THEORETICAL ASPECTS OF COMPUTER AND NETWORK SECURITY AS WELL AS HOW THAT THEORY IS APPLIED IN THE INTERNET. THIS KNOWLEDGE WILL HELP YOU IN DESIGNING AND DEVELOPING SECURE APPLICATIONS AND NETWORK PROTOCOLS AS WELL AS BUILDING SECURE NETWORKS.

c = ZIOL EGXKLT QODL ZG OFZKGRXET NGX ZG ZIT HKOFEOHSTL QFR ZTEIFOJXTL GY LTEXKOFU EGDHXZTKL QFR EGDHXZTK FTZVGKAL VOZI YGEXL GF OFZTKFTZ LTEXKOZN. ZIT EGXKLT OL TYYTEZOCTSN LHSOZ OFZG ZVG HQKZL. YOKLZ OFZKGRXEOFU ZIT ZITGKN GY EKNHZGUKQHIN OFESXROFU IGV DQFN ESQLLOEQS QFR HGHXSQK QSUGKOZIDL VGKA T.U. RTL, KLQ, ROUOZQS LOUFQZXKTL, QFR LTEGFR HKGCOROFU RTZQOSL GY KTQS OFZTKFTZ LTEXKOZN HKGZGEGSL, QSUGKOZIDL, QFR ZIKTQZL, T.U. OHLTE, COKXLTL, YOKTVQSSL. ITFET, NGX VOSS STQKF WGZI ZITGKTZOEQS QLHTEZL GY EGDHXZTK QFR FTZVGKA LTEXKOZN QL VTSS QL IGV ZIQZ ZITGKN OL QHHSOTR OF ZIT OFZTKFTZ. ZIOL AFGVSTRUT VOSS ITSH NGX OF RTLOUFOFU QFR RTCTSGHOFU LTEXKT QHHSOEQZOGFL QFR FTZVGKA HKGZGEGSL QL VTSS QL WXOSROFU LTEXKT FTZVGKAL.

THE UNIVERSITY of EDINBURGH

Computer Security

INFR10067

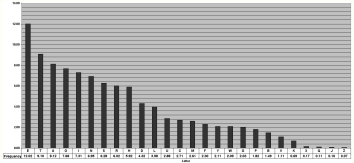
Fall 2025

Cryptography

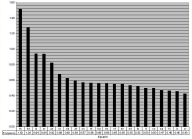
Quiz

Breaking the substitution cipher

- Key space size: $|\mathcal{K}| = 26! \ (\approx 2^{88})$ ⇒ brute force infeasible!
- Frequency analysis: exploit regularities of the language
  - Use frequency of letters in English text



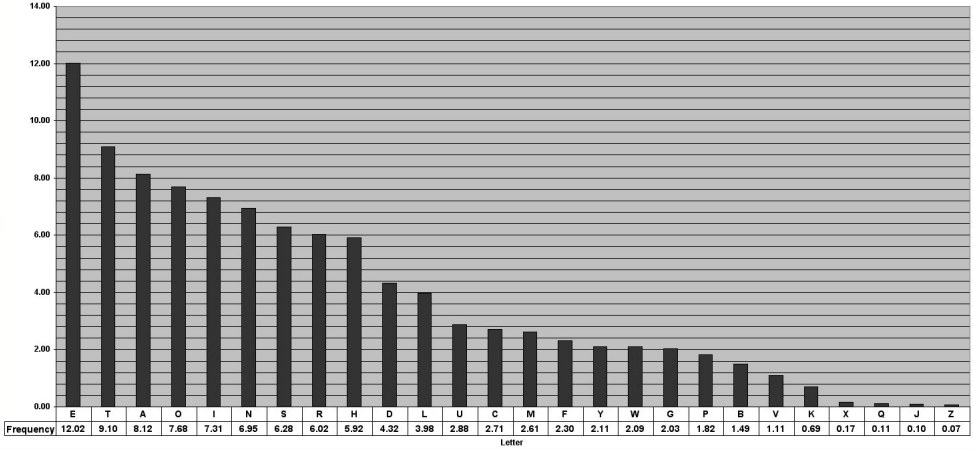
  - Use frequency of digrams in English text



  - Use frequency of trigrams in English text

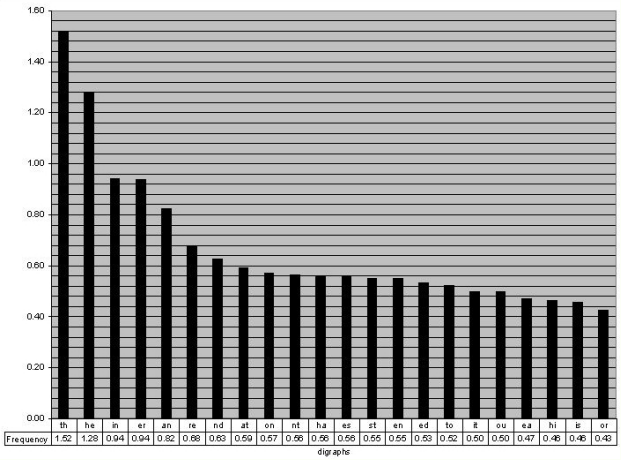
    the > and > ing
  - Use expected words

Frequency of letters



	E	T	A	O	I	N	S	R	H	D	L	U	C	M	F	Y	W	G	P	B	V	K	X	Q	J	Z
Frequency	12.02	9.10	8.12	7.68	7.31	6.95	6.28	6.02	5.92	4.32	3.98	2.88	2.71	2.61	2.30	2.11	2.09	2.03	1.82	1.49	1.11	0.69	0.17	0.11	0.10	0.07

Frequency of digrams



Breaking the substitution cipher: example

$\pi =$

c = ZIOL EGXKLT QODL ZG OFZKGRXET NGX ZG ZIT HKOFEOHSTL QFR ZTEIFOJXTL GY
LTEXKOFU EGDHXZTKL QFR EGDHXZTK FTZVGKAL VOZI YGEXL GF OFZTKFTZ
LTEXKOZN. ZIT EGXKLT OL TYYTEZOCTSN LHSOZ OFZG ZVG HQKZL. YOKLZ
OFZKGRXEOFU ZIT ZITGKN GY EKNHZGUKQHIN OFESXROFU IGV DQFN ESQLLOEQS
QFR HGHXSQK QSUGKOZIDL VGKA T.U. RTL, KLQ, ROUOZQS LOUFQZXKTL, QFR
LTEGFR HKGCOROFU RTZQOSL GY KTQS OFZTKFTZ LTEXKOZN HKGZGEGSL,
QSUGKOZIDL, QFR ZIKTQZL, T.U. OHLTE, COKXLTL, YOKTVQSSL. ITFET, NGX
VOSS STQKF WGZI ZITGKTZOEQS QLHTEZL GY EGDHXZTK QFR FTZVGKA
LTEXKOZN QL VTSS QL IGV ZIQZ ZITGKN OL QHHSOTR OF ZIT OFZTKFTZ. ZIOL
AFGVSTRUT VOSS ITSH NGX OF RTLOUFOFU QFR RTCTSGHOFU LTEXKT
QHHSOEQZOGFL QFR FTZVGKA HKGZGEGSL QL VTSS QL WXOSROFU LTEXKT
FTZVGKAL.

Breaking the substitution cipher: example

$\pi =$

c = ZIOL EGXKLT QODL ZG OFZKGRXET NGX ZG ZIT HKOFEOHSTL QFR ZTEIFOJXTL GY
LTEXKOFU EGDHXZTKL QFR EGDHXZTK FTZVGKAL VOZI YGEXL GF OFZTKFTZ
LTEXKOZN. ZIT EGXKLT OL TYYTEZOCTSN LHSOZ OFZG ZVG HQKZL. YOKLZ
OFZKGRXEOFU ZIT ZITGKN GY EKNHZGUKQHIN OFESXROFU IGV DQFN ESQLLOEQS
QFR HGHXSQK QSUGKOZIDL VGKA T.U. RTL, KLQ, ROUOZQS LOUFQZXKTL, QFR
LTEGFR HKGCOROFU RTZQOSL GY KTQS OFZTKFTZ LTEXKOZN HKGZGEGSL,
QSUGKOZIDL, QFR ZIKTQZL, T.U. OHLTE, COKXLTL, YOKTVQSSL. ITFET, NGX
VOSS STQKF WGZI ZITGKTZOEQS QLHTEZL GY EGDHXZTK QFR FTZVGKA
LTEXKOZN QL VTSS QL IGV ZIQZ ZITGKN OL QHHSOTR OF ZIT OFZTKFTZ. ZIOL
AFGVSTRUT VOSS ITSH NGX OF RTLOUFOFU QFR RTCTSGHOFU LTEXKT
QHHSOEQZOGFL QFR FTZVGKA HKGZGEGSL QL VTSS QL WXOSROFU LTEXKT
FTZVGKAL.

Most common letters in c: t > z > o > l

Breaking the substitution cipher: example

$\pi = \quad t \mapsto z \; e \mapsto t$

c = TIOL EGXKLE QODL TG OFTKGRXEE NGX TG TIE HKOFEOHSEL QFR TEEIFOJXEL GY
LEEXKOFU EGDHXTEKL QFR EGDHXTEK FETVGKAL VOTI YGEXL GF OFTEKFET
LEEXKOTN. TIE EGXKLE OL EYYEETOCESN LHSOT OFTG TVG HQKTL. YOKLT
OFTKGRXEOFU TIE TIEGKN GY EKNHTGUKQHIN OFESXROFU IGV DQFN ESQLLOEQS
QFR HGHXSQK QSUGKOTIDL VGKA E.U. REL, KLQ, ROUOTQS LOUFQTXKEL, QFR
LEEGFR HKGCOROFU RETQOSL GY KEQS OFTEKFET LEEXKOTN HKGTGEGSL,
QSUGKOTIDL, QFR TIKEQTL, E.U. OHLEE, COKXLEL, YOKEVQSSL. IEFEE, NGX
VOSS SEQKF WGTI TIEGKETOEQS QLHEETL GY EGDHXTEK QFR FETVGKA
LEEXKOTN QL VESS QL IGV TIQT TIEGKN OL QHHSOER OF TIE OFTEKFET. TIOL
AFGVSERUE VOSS IESH NGX OF RELOUFOFU QFR RECESGHOFU LEEXKE
QHHSOEQTOGFL QFR FETVGKA HKGTGEGSL QL VESS QL WXOSROFU LEEXKE
FETVGKAL.

Most common letters in c: t > z > ...

Breaking the substitution cipher: example

$\pi =$ $t \mapsto z$ $e \mapsto t$

c = TIOL EGXKLE QODL TG OFTKGRXEE NGX TG TIE HKOFEOHSEL QFR TEEIFOJXEL GY
LEEXKOFU EGDHXTEKL QFR EGDHXTEK FETVGKAL VOTI YGEXL GF OFTEKFET
LEEXKOTN. TIE EGXKLE OL EYYEETOCESN LHSOT OFTG TVG HQKTL. YOKLT
OFTKGRXEOFU TIE TIEGKN GY EKNHTGUKQHIN OFESXROFU IGV DQFN ESQLLOEQS
QFR HGHXSQK QSUGKOTIDL VGKA E.U. REL, KLQ, ROUOTQS LOUFQTXKEL, QFR
LEEGFR HKGCOROFU RETQOSL GY KEQS OFTEKFET LEEXKOTN HKGTGEGSL,
QSUGKOTIDL, QFR TIKEQTL, E.U. OHLEE, COKXLEL, YOKEVQSSL. IEFEE, NGX
VOSS SEQKF WGTI TIEGKETOEQS QLHEETL GY EGDHXTEK QFR FETVGKA
LEEXKOTN QL VESS QL IGV TIQT TIEGKN OL QHHSOER OF TIE OFTEKFET. TIOL
AFGVSERUE VOSS IESH NGX OF RELOUFOFU QFR RECESGHOFU LEEXKE
QHHSOEQTOGFL QFR FETVGKA HKGTGEGSL QL VESS QL WXOSROFU LEEXKE
FETVGKAL.

Most common digrams in c: of > zi > ...
t↦z suggests h↦i

Breaking the substitution cipher: example

$\pi = \quad t \mapsto z \; e \mapsto t \; h \mapsto i$

c = THOL EGXKLE QODL TG OFTKGRXEE NGX TG THE HKOFEOHSEL QFR TEEHFOJXEL GY
LEEXKOFU EGDHXTEKL QFR EGDHXTEK FETVGKAL VOTH YGEXL GF OFTEKFET
LEEXKOTN. THE EGXKLE OL EYYEETOCESN LHSOT OFTG TVG HQKTL. YOKLT
OFTKGRXEOFU THE THEGKN GY EKNHTGUKQHHN OFESXROFU HGV DQFN ESQLLOEQS
QFR HGHXSQK QSUGKOTHDL VGKA E.U. REL, KLQ, ROUOTQS LOUFQTXKEL, QFR
LEEGFR HKGCOROFU RETQOSL GY KEQS OFTEKFET LEEXKOTN HKGTGEGSL,
QSUGKOTHDL, QFR THKEQTL, E.U. OHLEE, COKXLEL, YOKEVQSSL. HEFEE, NGX
VOSS SEQKF WGTH THEGKETOEQS QLHEETL GY EGDHXTEK QFR FETVGKA
LEEXKOTN QL VESS QL HGV THQT THEGKN OL QHHSOER OF THE OFTEKFET. THOL
AFGVSERUE VOSS HESH NGX OF RELOUFOFU QFR RECESGHOFU LEEXKE
QHHSOEQTOGFL QFR FETVGKA HKGTGEGSL QL VESS QL WXOSROFU LEEXKE
FETVGKAL.

Most common digrams in c: of > zi > ...
we guess in↦of

Breaking the substitution cipher: example

$\pi = \quad t \mapsto z\, e \mapsto t\, h \mapsto i\, i \mapsto o\, n \mapsto f$

c = THIL EGXKLE QIDL TG INTKGRXEE NGX TG THE HKINEIHSEL QNR TEEHNIJXEL GY
LEEXKINU EGDHXTEKL QNR EGDHXTEK NETVGKAL VITH YGEXL GN INTEKNET
LEEXKITN. THE EGXKLE IL EYYEETICESN LHSIT INTG TVG HQKTL. YIKLT
INTKGRXEINU THE THEGKN GY EKNHTGUKQHHN INESXRINU HGV DQNN ESQLLIEQS
QNR HGHXSQK QSUGKITHDL VGKA E.U. REL, KLQ, RIUITQS LIUNQTXKEL, QNR
LEEGNR HKGCIRINU RETQISL GY KEQS INTEKNET LEEXKITN HKGTGEGSL,
QSUGKITHDL, QNR THKEQTL, E.U. IHLEE, CIKXLEL, YIKEVQSSL. HENEE, NGX
VISS SEQKN WGTH THEGKETIEQS QLHEETL GY EGDHXTEK QNR NETVGKA
LEEXKITN QL VESS QL HGV THQT THEGKN IL QHHSIER IN THE INTEKNET. THIL
ANGVSERUE VISS HESH NGX IN RELIUNINU QNR RECESGHINU LEEXKE
QHHSIEQTIGNL QNR NETVGKA HKGTGEGSL QL VESS QL WXISRINU LEEXKE
NETVGKAL.

Most common digrams in c: of > zi > ...

Breaking the substitution cipher: example

$\pi = \quad t \mapsto z\, e \mapsto t\, h \mapsto i\, i \mapsto o\, n \mapsto f$

c = THIL EGXKLE QIDL TG INTKGRXEE NGX TG THE HKINEIHSEL QNR TEEHNIJXEL GY
LEEXKINU EGDHXTEKL QNR EGDHXTEK NETVGKAL VITH YGEXL GN **INTEKNET**
LEEXKITN. THE EGXKLE IL EYYEETICESN LHSIT INTG TVG HQKTL. YIKLT
INTKGRXEINU THE THEGKN GY EKNHTGUKQHHN INESXRINU HGV DQNN ESQLLIEQS
QNR HGHXSQK QSUGKITHDL VGKA E.U. REL, KLQ, RIUITQS LIUNQTXKEL, QNR
LEEGNR HKGCIRINU RETQISL GY KEQS INTEKNET LEEXKITN HKGTGEGSL,
QSUGKITHDL, QNR THKEQTL, E.U. IHLEE, CIKXLEL, YIKEVQSSL. HENEE, NGX
VISS SEQKN WGTH THEGKETIEQS QLHEETL GY EGDHXTEK QNR NETVGKA
LEEXKITN QL VESS QL HGV THQT THEGKN IL QHHSIER IN THE INTEKNET. THIL
ANGVSERUE VISS HESH NGX IN RELIUNINU QNR RECESGHINU LEEXKE
QHHSIEQTIGNL QNR NETVGKA HKGTGEGSL QL VESS QL WXISRINU LEEXKE
NETVGKAL.

We identify in c the word **INTEKNET**
suggests r↦k

Breaking the substitution cipher: example

$\pi = \quad t \mapsto z\, e \mapsto t\, h \mapsto i\, i \mapsto o\, n \mapsto f\, r \mapsto k$

c = THIL EGXRLE QIDL TG INTRGRXEE NGX TG THE HRINEIHSEL QNR TEEHNIJXEL GY
LEEXRINU EGDHXTERL QNR EGDHXTER NETVGRAL VITH YGEXL GN **INTERNET**
LEEXRITN. THE EGXRLE IL EYYEETICESN LHSIT INTG TVG HQRTL. YIRLT
INTRGRXEINU THE THEGRN GY ERNHTGURQHHN INESXRINU HGV DQNN ESQLLIEQS
QNR HGHXSQR QSUGRITHDL VGRA E.U. REL, RLQ, RIUITQS LIUNQTXREL, QNR
LEEGNR HRGCIRINU RETQISL GY REQS INTERNET LEEXRITN HRGTGEGSL,
QSUGRITHDL, QNR THREQTL, E.U. IHLEE, CIRXLEL, YIREVQSSL. HENEE, NGX
VISS SEQRN WGTH THEGRETIEQS QLHEETL GY EGDHXTER QNR NETVGRA
LEEXRITN QL VESS QL HGV THQT THEGRN IL QHHSIER IN THE INTERNET. THIL
ANGVSERUE VISS HESH NGX IN RELIUNINU QNR RECESGHINU LEEXYE
QHHSIEQTIGNL QNR NETVGRA HRGTGEGSL QL VESS QL WXISRINU LEEXRE
NETVGRAL.

We identify in c the word **INTEKNET**

Breaking the substitution cipher: example

$\pi = \quad t \mapsto z\, e \mapsto t\, h \mapsto i\, i \mapsto o\, n \mapsto f\, r \mapsto k$

c = THIL EGXRLE QIDL TG INTRGRXEE NGX TG THE HRINEIHSEL QNR TEEHNIJXEL GY
LEEXRINU EGDHXTERL QNR EGDHXTER NETVGRAL VITH YGEXL GN INTERNET
LEEXRITN. THE EGXRLE IL EYYEETICESN LHSIT INTG TVG HQRTL. YIRLT
INTRGRXEINU THE THEGRN GY ERNHTGURQHHN INESXRINU HGV DQNN ESQLLIEQS
QNR HGHXSQR QSUGRITHDL VGRA E.U. REL, RLQ, RIUITQS LIUNQTXREL, QNR
LEEGNR HRGCIRINU RETQISL GY REQS INTERNET LEEXRITN HRGTGEGSL,
QSUGRITHDL, QNR THREQTL, E.U. IHLEE, CIRXLEL, YIREVQSSL. HENEE, NGX
VISS SEQRN WGTH THEGRETIEQS QLHEETL GY EGDHXTER QNR NETVGRA
LEEXRITN QL VESS QL HGV THQT THEGRN IL QHHSIER IN THE INTERNET. THIL
ANGVSERUE VISS HESH NGX IN RELIUNINU QNR RECESGHINU LEEXYE
QHHSIEQTIGNL QNR NETVGRA HRGTGEGSL QL VESS QL WXISRINU LEEXRE
NETVGRAL.

The first word is THIL
suggests s↦l

Breaking the substitution cipher: example

$\pi = \quad t \mapsto z\, e \mapsto t\, h \mapsto i\, i \mapsto o\, n \mapsto f\, r \mapsto k\, s \mapsto l$

c = THIS EGXRSE QIDS TG INTRGRXEE NGX TG THE HRINEIHSES QNR TEEHNIJXES GY
SEEXRINU EGDHXTERS QNR EGDHXTER NETVGRAS VITH YGEXS GN INTERNET
SEEXRITN. THE EGXRSE IS EYYEETICESN SHSIT INTG TVG HQRTS. YIRST
INTRGRXEINU THE THEGRN GY ERNHTGURQHHN INESXRINU HGV DQNN ESQSSIEQS
QNR HGHXSQR QSUGRITHDS VGRA E.U. RES, RSQ, RIUITQS SIUNQTXRES, QNR
SEEGNR HRGCIRINU RETQISS GY REQS INTERNET SEEXRITN HRGTGEGSS,
QSUGRITHDS, QNR THREQTS, E.U. IHSEE, CIRXSES, YIREVQSSS. HENEE, NGX
VISS SEQRN WGTH THEGRETIEQS QSHEETS GY EGDHXTER QNR NETVGRA
SEEXRITN QS VESS QS HGV THQT THEGRN IS QHHSIER IN THE INTERNET. THIS
ANGVSERUE VISS HESH NGX IN RESIUNINU QNR RECESGHINU SEEXRE
QHHSIEQTIGNS QNR NETVGRA HRGTGEGSS QS VESS QS WXISRINU SEEXRE
NETVGRAS.

The first word is THIL

Breaking the substitution cipher: example

$\pi = \quad t \mapsto z\, e \mapsto t\, h \mapsto i\, i \mapsto o\, n \mapsto f\, r \mapsto k\, s \mapsto l$

c = THIS EGXRSE QIDS TG INTRGRXEE NGX TG THE HRINEIHSES QNR TEEHNIJXES GY
SEEXRINU EGDHXTERS QNR EGDHXTER NETVGRAS VITH YGEXS GN INTERNET
SEEXRITN. THE EGXRSE IS EYYEETICESN SHSIT INTG TVG HQRTS. YIRST
INTRGRXEINU THE THEGRN GY ERNHTGURQHHN INESXRINU HGV DQNN ESQSSIEQS
QNR HGHXSQR QSUGRITHDS VGRA E.U. RES, RSQ, RIUITQS SIUNQTXRES, QNR
SEEGNR HRGCIRINU RETQISS GY REQS INTERNET SEEXRITN HRGTGEGSS,
QSUGRITHDS, QNR THREQTS, E.U. IHSEE, CIRXSES, YIREVQSSS. HENEE, NGX
VISS SEQRN WGTH THEGRETIEQS QSHEETS GY EGDHXTER QNR NETVGRA
SEEXRITN QS VESS QS HGV THQT THEGRN IS QHHSIER IN THE INTERNET. THIS
ANGVSERUE VISS HESH NGX IN RESIUNINU QNR RECESGHINU SEEXRE
QHHSIEQTIGNS QNR NETVGRA HRGTGEGSS QS VESS QS WXISRINU SEEXRE
NETVGRAS.

Going back to letter frequency and a few more guesses!!

Breaking the substitution cipher: example

$\pi =$ $a \mapsto q\ b \mapsto w\ c \mapsto e\ d \mapsto r\ e \mapsto t\ f \mapsto y\ g \mapsto u\ h \mapsto i\ i \mapsto o\ j \mapsto m\ k \mapsto a\ l \mapsto s$
$m \mapsto d\ n \mapsto f\ o \mapsto g\ p \mapsto h\ q \mapsto j\ r \mapsto k\ s \mapsto l\ t \mapsto z\ u \mapsto x\ v \mapsto c\ w \mapsto v\ x \mapsto b$
$y \mapsto n\ z \mapsto p$

m = THIS COURSE AIMS TO INTRODUCE YOU TO THE PRINCIPLES AND TECHNIQUES OF SECURING COMPUTERS AND COMPUTER NETWORKS WITH FOCUS ON INTERNET SECURITY. THE COURSE IS EFFECTIVELY SPLIT INTO TWO PARTS. FIRST INTRODUCING THE THEORY OF CRYPTOGRAPHY INCLUDING HOW MANY CLASSICAL AND POPULAR ALGORITHMS WORK E.G. DES, RSA, DIGITAL SIGNATURES, AND SECOND PROVIDING DETAILS OF REAL INTERNET SECURITY PROTOCOLS, ALGORITHMS, AND THREATS, E.G. IPSEC, VIRUSES, FIREWALLS. HENCE, YOU WILL LEARN BOTH THEORETICAL ASPECTS OF COMPUTER AND NETWORK SECURITY AS WELL AS HOW THAT THEORY IS APPLIED IN THE INTERNET. THIS KNOWLEDGE WILL HELP YOU IN DESIGNING AND DEVELOPING SECURE APPLICATIONS AND NETWORK PROTOCOLS AS WELL AS BUILDING SECURE NETWORKS.

Going back to letter frequency and a few more guesses!!

A better substitution cipher: The One-Time Pad (OTP)

- $\mathcal{M} = \mathcal{C} = \mathcal{K} = \{0,1\}^n$
- Encryption: $\forall k \in \mathcal{K}. \forall m \in \mathcal{M}. E(k,m) = k \oplus m$

$$\begin{array}{rcccccccc} k & = & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ m & = & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ \hline c & = & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \end{array}$$

- Decryption: $\forall k \in \mathcal{K}. \forall c \in \mathcal{C}. D(k,c) = k \oplus c$

$$\begin{array}{rcccccccc} k & = & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ c & = & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ \hline m & = & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{array}$$

- Correctness: $D(k, E(k,m)) = k \oplus (k \oplus m) = m$

Perfect secrecy

Definition

A cipher (E, D) over $(\mathcal{M}, \mathcal{C}, \mathcal{K})$ satisfies perfect secrecy if for all messages $m_1, m_2 \in \mathcal{M}$ of same length ($|m_1| = |m_2|$), and for all ciphertexts $c \in \mathcal{C}$

$$|Pr(E(k, m_1) = c) - Pr(E(k, m_2) = c)| \leq \epsilon$$

where $k \xleftarrow{r} \mathcal{K}$ and ϵ is some "negligible quantity".

OTP satisfies perfect secrecy

Theorem (Shannon 1949)

The One-Time Pad satisfies perfect secrecy

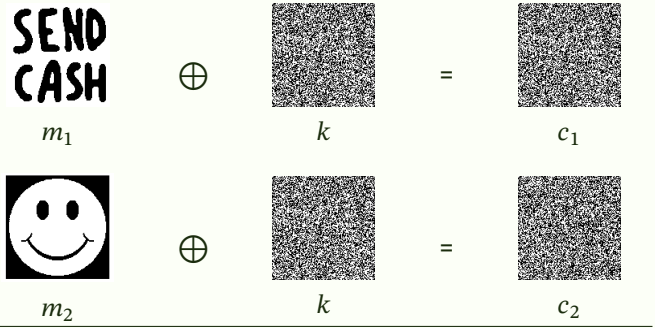
Proof: We first note that for all messages $m \in \mathcal{M}$ and all ciphertexts $c \in \mathcal{C}$

$$\begin{aligned} Pr(E(k,m) = c) &= \frac{\#\{k \in \mathcal{K} \colon k \oplus m = c\}}{\#\mathcal{K}} \\ &= \frac{\#\{k \in \mathcal{K} \colon k = m \oplus c\}}{\#\mathcal{K}} \\ &= \frac{1}{\#\mathcal{K}} \end{aligned}$$

where $k \xleftarrow{r} \mathcal{K}$.
Thus, for all messages $m_1, m_2 \in \mathcal{M}$, and for all ciphertexts $c \in \mathcal{C}$

$$|Pr(E(k,m_1) = c) - Pr(E(k,m_2) = c)| \leq \left| \frac{1}{\#\mathcal{K}} - \frac{1}{\#\mathcal{K}} \right| = 0$$

Two-time pad attacks



Source: Cryptosmith and David Lowry-Duda

Limitations of OTP

- Key-length!
  - The key should be as long as the plaintext
- Getting true randomness!
  - The key should not be guessable from an attacker
  - If the key is not truly random, frequency analysis might again be possible
- Perfect secrecy does not capture all possible attacks
  - OTP is subject to two-time pad attacks
    given $m_1 \oplus k$ and $m_2 \oplus k$, we can compute $m_1 \oplus m_2 = (m_1 \oplus k) \oplus (m_2 \oplus k)$
    English has enough redundancy s.t. $m_1 \oplus m_2 \rightarrow m_1, m_2$
  - OTP is malleable
    given the ciphertext $c = E(k, m)$ with $m = to\ bob : m_0$, it is possible to compute the ciphertext $c' = E(k, m')$ with $m' = to\ eve : m_0$
    $c' := c \oplus "to\ bob : 00 \ldots 00" \oplus "to\ eve : 00 \ldots 00"$

Concluding remark

The confidentiality problem is now reduced to a key management problem:

- Where are keys generated?
- How are keys generated?
- Where are keys stored?
- Where are the keys actually used?
- How are key revoked and replaced?

THE UNIVERSITY of EDINBURGH

Computer Security

INFR10067

Fall 2025

Cryptography

Symmetric encryption

Markulf Kohlweiss
School of Informatics
University of Edinburgh

Recap

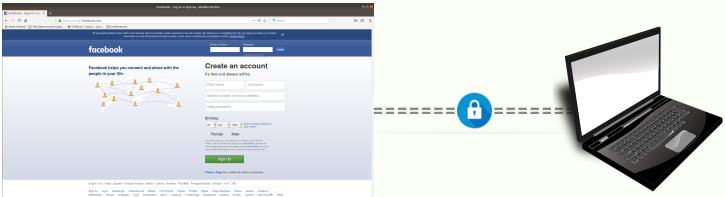
Action items:

- ✓ Request: Upload slides in advance (note that they might change before, and even after the lecture)
- ✓ Question: When will exam schedule become available? 1st of Nov, exams are 8-19th Dec.
- ✓ Question: When are the office hours? 11:15 Mon and Fri, talk with me after lecture.
- ? Request: Learn about Post-Quantum Cryptography. What are the biggest open problems in cryptography?
- ? Request: Practical examples from case studies and news.

We made it till Kerckhoff's principle last time.

Goal: confidentiality

- Secure communications



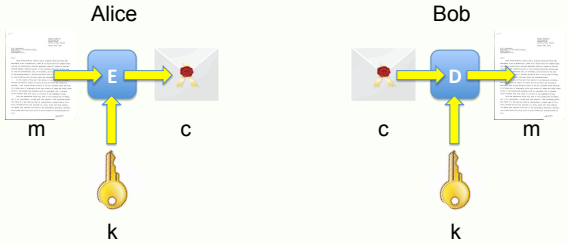
- File protection

Symmetric encryption schemes

A symmetric cipher consists of two algorithms

- encryption algorithm $E : \mathcal{K} \times \mathcal{M} \to \mathcal{C}$
- decryption algorithm $D : \mathcal{K} \times \mathcal{C} \to \mathcal{M}$

st. $\forall k \in \mathcal{K}$, and $\forall m \in \mathcal{M}, D(k, E(k, m)) = m$



- same key k to encrypt and decrypt
- the key k is secret: only known to Alice and Bob

What is a good encryption scheme?

An encryption scheme is secure against a given adversary, if this adversary cannot

- recover the secret key k
- recover the plaintext m underlying a ciphertext c
- recover any bits of the plaintext m underlying a ciphertext c
- ...

Kerckhoff's principle

The architecture and design of a security system/mechanism should be made public

No security through obscurity!

- The encryption (E) and decryption (D) algorithms are public
- The security relies entirely on the secrecy of the key

Open design allows for a system to be scrutinised by many users, white hat hackers, academics, *etc.*

⇝ early discovery and corrections of flaws/vulnerabilities

Adversary's capabilities

- A cryptographic scheme is secure under some assumptions about the **power of the attacker** and they **kind of attacks** it can perform

The attacker know the encryption/decryption algorithms but may have access to :

- unlimited or realistic ($\leq 2^{80}$) computational power, or polynomial in key size
- Ciphertext only attack - some ciphertexts c_1, ..., c_n
- Known plaintext attack some plaintext/ciphertext pairs (m_1, c_1), ..., (m_n, c_n) st. $c_i = E(k, m_i)$)
- Chosen plaintext attack - he has access to an encryption oracle - can maybe trick a user to encrypt messages m_1, ..., m_n of his choice
- Chosen ciphertext attack - he has access to a decryption oracle - can maybe trick a user to decrypt ciphertexts c_1, ..., c_n of his choice

Chosen plaintext attack – Battle of Midway

Yorktown shortly after being hit by three Japanese bombs

Brute-force attack - attack on all schemes

- Try all possible keys $k \in \mathcal{K}$ - requires some knowledge about the structure of plaintext

- Making exhaustive search unfeasible:
  - $\mathcal{K}$ should be sufficiently large, *i.e.* keys should be sufficiently long
  - Keys should be sampled uniformly at random from $\mathcal{K}$

A simple scheme: the substitution cipher

- shared secret: a permutation π of the set of characters

$$\pi = \begin{array}{llllllllll} a \mapsto q & b \mapsto w & c \mapsto e & d \mapsto r & e \mapsto t & f \mapsto y & g \mapsto u & h \mapsto i & i \mapsto o \\ j \mapsto m & k \mapsto a & l \mapsto s & m \mapsto d & n \mapsto f & o \mapsto g & p \mapsto h & q \mapsto j & r \mapsto k \\ s \mapsto l & t \mapsto z & u \mapsto x & v \mapsto c & w \mapsto v & x \mapsto b & y \mapsto n & z \mapsto p \end{array}$$

- Encryption: apply π to each character of the plaintext

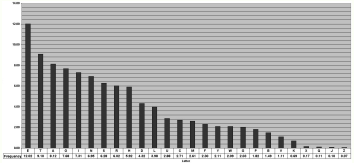
$$E(\pi, p_1 \dots p_n) = \pi(p_1) \dots \pi(p_n)$$

- Decryption: apply π^{-1} to each character of the plaintext

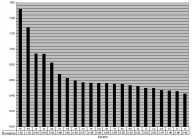
$$D(\pi, c_1 \dots c_n) = \pi^{-1}(c_1) \dots \pi^{-1}(c_n)$$

Breaking the substitution cipher

- Key space size: $|\mathcal{K}| = 26! \ (\approx 2^{88})$ ⇒ brute force infeasible!
- Frequency analysis: exploit regularities of the language
  - Use frequency of letters in English text



  - Use frequency of digrams in English text

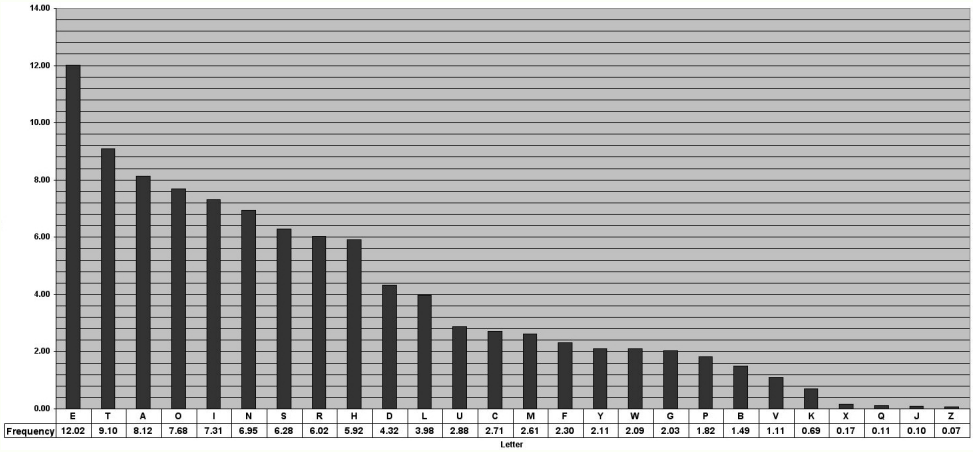


  - Use frequency of trigrams in English text

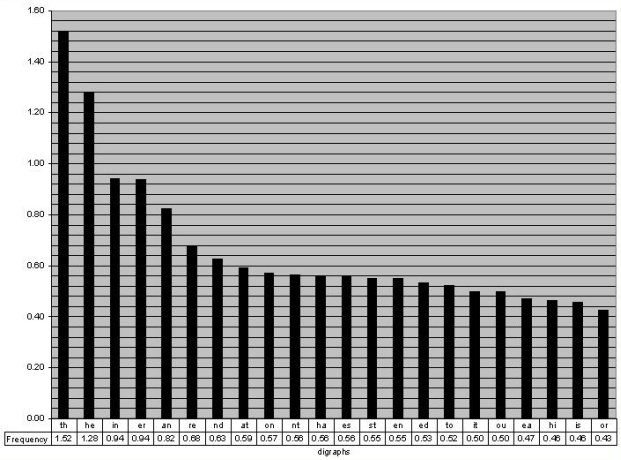
the > and > ing

  - Use expected words

Frequency of letters



Frequency of digrams



Breaking the substitution cipher: example

$\pi =$

c = ZIOL EGXKLT QODL ZG OFZKGRXET NGX ZG ZIT HKOFEOHSTL QFR ZTEIFOJXTL GY LTEXKOFU EGDHXZTKL QFR EGDHXZTK FTZVGKAL VOZI YGEXL GF OFZTKFTZ LTEXKOZN. ZIT EGXKLT OL TYYTEZOCTSN LHSOZ OFZG ZVG HQKZL. YOKLZ OFZKGRXEOFU ZIT ZITGKN GY EKNHZGUKQHIN OFESXROFU IGV DQFN ESQLLOEQS QFR HGHXSQK QSUGKOZIDL VGKA T.U. RTL, KLQ, ROUOZQS LOUFQZXKTL, QFR LTEGFR HKGCOROFU RTZQOSL GY KTQS OFZTKFTZ LTEXKOZN HKGZGEGSL, QSUGKOZIDL, QFR ZIKTQZL, T.U. OHLTE, COKXLTL, YOKTVQSSL. ITFET, NGX VOSS STQKF WGZI ZITGKTZOEQS QLHTEZL GY EGDHXZTK QFR FTZVGKA LTEXKOZN QL VTSS QL IGV ZIQZ ZITGKN OL QHHSOTR OF ZIT OFZTKFTZ. ZIOL AFGVSTRUT VOSS ITSH NGX OF RTLOUFOFU QFR RTCTSGHOFU LTEXKT QHHSOEQZOGFL QFR FTZVGKA HKGZGEGSL QL VTSS QL WXOSROFU LTEXKT FTZVGKAL.

Breaking the substitution cipher: example

π =

c = ZIOL EGXKLT QODL ZG OFZKGRXET NGX ZG ZIT HKOFEOHSTL QFR ZTEIFOJXTL GY LTEXKOFU EGDHXZTKL QFR EGDHXZTK FTZVGKAL VOZI YGEXL GF OFZTKFTZ LTEXKOZN. ZIT EGXKLT OL TYYTEZOCTSN LHSOZ OFZG ZVG HQKZL. YOKLZ OFZKGRXEOFU ZIT ZITGKN GY EKNHZGUKQHIN OFESXROFU IGV DQFN ESQLLOEQS QFR HGHXSQK QSUGKOZIDL VGKA T.U. RTL, KLQ, ROUOZQS LOUFQZXKTL, QFR LTEGFR HKGCOROFU RTZQOSL GY KTQS OFZTKFTZ LTEXKOZN HKGZGEGSL, QSUGKOZIDL, QFR ZIKTQZL, T.U. OHLTE, COKXLTL, YOKTVQSSL. ITFET, NGX VOSS STQKF WGZI ZITGKTZOEQS QLHTEZL GY EGDHXZTK QFR FTZVGKA LTEXKOZN QL VTSS QL IGV ZIQZ ZITGKN OL QHHSOTR OF ZIT OFZTKFTZ. ZIOL AFGVSTRUT VOSS ITSH NGX OF RTLOUFOFU QFR RTCTSGHOFU LTEXKT QHHSOEQZOGFL QFR FTZVGKA HKGZGEGSL QL VTSS QL WXOSROFU LTEXKT FTZVGKAL.

Most common letters in c: t > z > o > l

Breaking the substitution cipher: example

$\pi = \quad e \mapsto t \quad t \mapsto z$

c = TIOL EGXKLE QODL TG OFTKGRXEE NGX TG TIE HKOFEOHSEL QFR TEEIFOJXEL GY
LEEXKOFU EGDHXTEKL QFR EGDHXTEK FETVGKAL VOTI YGEXL GF OFTEKFET
LEEXKOTN. TIE EGXKLE OL EYYEETOCESN LHSOT OFTG TVG HQKTL. YOKLT
OFTKGRXEOFU TIE TIEGKN GY EKNHTGUKQHIN OFESXROFU IGV DQFN ESQLLOEQS
QFR HGHXSQK QSUGKOTIDL VGKA E.U. REL, KLQ, ROUOTQS LOUFQTXKEL, QFR
LEEGFR HKGCOROFU RETQOSL GY KEQS OFTEKFET LEEXKOTN HKGTGEGSL,
QSUGKOTIDL, QFR TIKEQTL, E.U. OHLEE, COKXLEL, YOKEVQSSL. IEFEE, NGX
VOSS SEQKF WGTI TIEGKETOEQS QLHEETL GY EGDHXTEK QFR FETVGKA
LEEXKOTN QL VESS QL IGV TIQT TIEGKN OL QHHSOER OF TIE OFTEKFET. TIOL
AFGVSERUE VOSS IESH NGX OF RELOUFOFU QFR RECESGHOFU LEEXKE
QHHSOEQTOGFL QFR FETVGKA HKGTGEGSL QL VESS QL WXOSROFU LEEXKE
FETVGKAL.

Most common letters in c: t > z > ...

Breaking the substitution cipher: example

$\pi =$ $e \mapsto t$ $t \mapsto z$

c = TIOL EGXKLE QODL TG OFTKGRXEE NGX TG TIE HKOFEOHSEL QFR TEEIFOJXEL GY
LEEXKOFU EGDHXTEKL QFR EGDHXTEK FETVGKAL VOTI YGEXL GF OFTEKFET
LEEXKOTN. TIE EGXKLE OL EYYEETOCESN LHSOT OFTG TVG HQKTL. YOKLT
OFTKGRXEOFU TIE TIEGKN GY EKNHTGUKQHIN OFESXROFU IGV DQFN ESQLLOEQS
QFR HGHXSQK QSUGKOTIDL VGKA E.U. REL, KLQ, ROUOTQS LOUFQTXKEL, QFR
LEEGFR HKGCOROFU RETQOSL GY KEQS OFTEKFET LEEXKOTN HKGTGEGSL,
QSUGKOTIDL, QFR TIKEQTL, E.U. OHLEE, COKXLEL, YOKEVQSSL. IEFEE, NGX
VOSS SEQKF WGTI TIEGKETOEQS QLHEETL GY EGDHXTEK QFR FETVGKA
LEEXKOTN QL VESS QL IGV TIQT TIEGKN OL QHHSOER OF TIE OFTEKFET. TIOL
AFGVSERUE VOSS IESH NGX OF RELOUFOFU QFR RECESGHOFU LEEXKE
QHHSOEQTOGFL QFR FETVGKA HKGTGEGSL QL VESS QL WXOSROFU LEEXKE
FETVGKAL.

Most common digrams in c: of > zi > ...
t↦z suggests h↦i

Breaking the substitution cipher: example

$\pi =$ $e \mapsto t$ $t \mapsto z$ $h \mapsto i$

c = THOL EGXKLE QODL TG OFTKGRXEE NGX TG THE HKOFEOHSEL QFR TEEHFOJXEL GY
LEEXKOFU EGDHXTEKL QFR EGDHXTEK FETVGKAL VOTH YGEXL GF OFTEKFET
LEEXKOTN. THE EGXKLE OL EYYEETOCESN LHSOT OFTG TVG HQKTL. YOKLT
OFTKGRXEOFU THE THEGKN GY EKNHTGUKQHHN OFESXROFU HGV DQFN ESQLLOEQS
QFR HGHXSQK QSUGKOTHDL VGKA E.U. REL, KLQ, ROUOTQS LOUFQTXKEL, QFR
LEEGFR HKGCOROFU RETQOSL GY KEQS OFTEKFET LEEXKOTN HKGTGEGSL,
QSUGKOTHDL, QFR THKEQTL, E.U. OHLEE, COKXLEL, YOKEVQSSL. HEFEE, NGX
VOSS SEQKF WGTH THEGKETOEQS QLHEETL GY EGDHXTEK QFR FETVGKA
LEEXKOTN QL VESS QL HGV THQT THEGKN OL QHHSOER OF THE OFTEKFET. THOL
AFGVSERUE VOSS HESH NGX OF RELOUFOFU QFR RECESGHOFU LEEXKE
QHHSOEQTOGFL QFR FETVGKA HKGTGEGSL QL VESS QL WXOSROFU LEEXKE
FETVGKAL.

Most common digrams in c: of > zi > ...
we guess in↦of

Breaking the substitution cipher: example

$\pi = \quad e \mapsto t \quad t \mapsto z \quad h \mapsto i \quad i \mapsto o \quad n \mapsto f$

c = THIL EGXKLE QIDL TG INTKGRXEE NGX TG THE HKINEIHSEL QNR TEEHNIJXEL GY
LEEXKINU EGDHXTEKL QNR EGDHXTEK NETVGKAL VITH YGEXL GN INTEKNET
LEEXKITN. THE EGXKLE IL EYYEETICESN LHSIT INTG TVG HQKTL. YIKLT
INTKGRXEINU THE THEGKN GY EKNHTGUKQHHN INESXRINU HGV DQNN ESQLLIEQS
QNR HGHXSQK QSUGKITHDL VGKA E.U. REL, KLQ, RIUITQS LIUNQTXKEL, QNR
LEEGNR HKGCIRINU RETQISL GY KEQS INTEKNET LEEXKITN HKGTGEGSL,
QSUGKITHDL, QNR THKEQTL, E.U. IHLEE, CIKXLEL, YIKEVQSSL. HENEE, NGX
VISS SEQKN WGTH THEGKETIEQS QLHEETL GY EGDHXTEK QNR NETVGKA
LEEXKITN QL VESS QL HGV THQT THEGKN IL QHHSIER IN THE INTEKNET. THIL
ANGVSERUE VISS HESH NGX IN RELIUNINU QNR RECESGHINU LEEXKE
QHHSIEQTIGNL QNR NETVGKA HKGTGEGSL QL VESS QL WXISRINU LEEXKE
NETVGKAL.

Most common digrams in c: of > zi > ...

Breaking the substitution cipher: example

$\pi =$ $e \mapsto t$ $t \mapsto z$ $h \mapsto i$ $i \mapsto o$ $n \mapsto f$

c = THIL EGXKLE QIDL TG INTKGRXEE NGX TG THE HKINEIHSEL QNR TEEHNIJXEL GY
LEEXKINU EGDHXTEKL QNR EGDHXTEK NETVGKAL VITH YGEXL GN **INTEKNET**
LEEXKITN. THE EGXKLE IL EYYEETICESN LHSIT INTG TVG HQKTL. YIKLT
INTKGRXEINU THE THEGKN GY EKNHTGUKQHHN INESXRINU HGV DQNN ESQLLIEQS
QNR HGHXSQK QSUGKITHDL VGKA E.U. REL, KLQ, RIUITQS LIUNQTXKEL, QNR
LEEGNR HKGCIRINU RETQISL GY KEQS INTEKNET LEEXKITN HKGTGEGSL,
QSUGKITHDL, QNR THKEQTL, E.U. IHLEE, CIKXLEL, YIKEVQSSL. HENEE, NGX
VISS SEQKN WGTH THEGKETIEQS QLHEETL GY EGDHXTEK QNR NETVGKA
LEEXKITN QL VESS QL HGV THQT THEGKN IL QHHSIER IN THE INTEKNET. THIL
ANGVSERUE VISS HESH NGX IN RELIUNINU QNR RECESGHINU LEEXKE
QHHSIEQTIGNL QNR NETVGKA HKGTGEGSL QL VESS QL WXISRINU LEEXKE
NETVGKAL.

We identify in c the word **INTEKNET**
suggests r↦k

Breaking the substitution cipher: example

$\pi = \quad e \mapsto t \quad t \mapsto z \quad h \mapsto i \quad i \mapsto o \quad n \mapsto f \quad r \mapsto k$

c = THIL EGXRLE QIDL TG INTRGRXEE NGX TG THE HRINEIHSEL QNR TEEHNIJXEL GY
LEEXRINU EGDHXTERL QNR EGDHXTER NETVGRAL VITH YGEXL GN **INTERNET**
LEEXRITN. THE EGXRLE IL EYYEETICESN LHSIT INTG TVG HQRTL. YIRLT
INTRGRXEINU THE THEGRN GY ERNHTGURQHHN INESXRINU HGV DQNN ESQLLIEQS
QNR HGHXSQR QSUGRITHDL VGRA E.U. REL, RLQ, RIUITQS LIUNQTXREL, QNR
LEEGNR HRGCIRINU RETQISL GY REQS INTERNET LEEXRITN HRGTGEGSL,
QSUGRITHDL, QNR THREQTL, E.U. IHLEE, CIRXLEL, YIREVQSSL. HENEE, NGX
VISS SEQRN WGTH THEGRETIEQS QLHEETL GY EGDHXTER QNR NETVGRA
LEEXRITN QL VESS QL HGV THQT THEGRN IL QHHSIER IN THE INTERNET. THIL
ANGVSERUE VISS HESH NGX IN RELIUNINU QNR RECESGHINU LEEXYE
QHHSIEQTIGNL QNR NETVGRA HRGTGEGSL QL VESS QL WXISRINU LEEXRE
NETVGRAL.

We identify in c the word **INTEKNET**

Breaking the substitution cipher: example

$\pi =$ $e \mapsto t$ $t \mapsto z$ $h \mapsto i$ $i \mapsto o$ $n \mapsto f$ $r \mapsto k$

c = THIL EGXRLE QIDL TG INTRGRXEE NGX TG THE HRINEIHSEL QNR TEEHNIJXEL GY
LEEXRINU EGDHXTERL QNR EGDHXTER NETVGRAL VITH YGEXL GN INTERNET
LEEXRITN. THE EGXRLE IL EYYEETICESN LHSIT INTG TVG HQRTL. YIRLT
INTRGRXEINU THE THEGRN GY ERNHTGURQHHN INESXRINU HGV DQNN ESQLLIEQS
QNR HGHXSQR QSUGRITHDL VGRA E.U. REL, RLQ, RIUITQS LIUNQTXREL, QNR
LEEGNR HRGCIRINU RETQISL GY REQS INTERNET LEEXRITN HRGTGEGSL,
QSUGRITHDL, QNR THREQTL, E.U. IHLEE, CIRXLEL, YIREVQSSL. HENEE, NGX
VISS SEQRN WGTH THEGRETIEQS QLHEETL GY EGDHXTER QNR NETVGRA
LEEXRITN QL VESS QL HGV THQT THEGRN IL QHHSIER IN THE INTERNET. THIL
ANGVSERUE VISS HESH NGX IN RELIUNINU QNR RECESGHINU LEEXYE
QHHSIEQTIGNL QNR NETVGRA HRGTGEGSL QL VESS QL WXISRINU LEEXRE
NETVGRAL.

The first word is THIL
suggests s↦l

Breaking the substitution cipher: example

$\pi =$ $e \mapsto t$ $t \mapsto z$ $h \mapsto i$ $i \mapsto o$ $n \mapsto f$ $r \mapsto k$ $s \mapsto l$

c = THIS EGXRSE QIDS TG INTRGRXEE NGX TG THE HRINEIHSES QNR TEEHNIJXES GY
SEEXRINU EGDHXTERS QNR EGDHXTER NETVGRAS VITH YGEXS GN INTERNET
SEEXRITN. THE EGXRSE IS EYYEETICESN SHSIT INTG TVG HQRTS. YIRST
INTRGRXEINU THE THEGRN GY ERNHTGURQHHN INESXRINU HGV DQNN ESQSSIEQS
QNR HGHXSQR QSUGRITHDS VGRA E.U. RES, RSQ, RIUITQS SIUNQTXRES, QNR
SEEGNR HRGCIRINU RETQISS GY REQS INTERNET SEEXRITN HRGTGEGSS,
QSUGRITHDS, QNR THREQTS, E.U. IHSEE, CIRXSES, YIREVQSSS. HENEE, NGX
VISS SEQRN WGTH THEGRETIEQS QSHEETS GY EGDHXTER QNR NETVGRA
SEEXRITN QS VESS QS HGV THQT THEGRN IS QHHSIER IN THE INTERNET. THIS
ANGVSERUE VISS HESH NGX IN RESIUNINU QNR RECESGHINU SEEXRE
QHHSIEQTIGNS QNR NETVGRA HRGTGEGSS QS VESS QS WXISRINU SEEXRE
NETVGRAS.

The first word is THIL

Breaking the substitution cipher: example

$\pi =$ $e \mapsto t$ $t \mapsto z$ $h \mapsto i$ $i \mapsto o$ $n \mapsto f$ $r \mapsto k$ $s \mapsto l$

c = THIS EGXRSE QIDS TG INTRGRXEE NGX TG THE HRINEIHSES QNR TEEHNIJXES GY
SEEXRINU EGDHXTERS QNR EGDHXTER NETVGRAS VITH YGEXS GN INTERNET
SEEXRITN. THE EGXRSE IS EYYEETICESN SHSIT INTG TVG HQRTS. YIRST
INTRGRXEINU THE THEGRN GY ERNHTGURQHHN INESXRINU HGV DQNN ESQSSIEQS
QNR HGHXSQR QSUGRITHDS VGRA E.U. RES, RSQ, RIUITQS SIUNQTXRES, QNR
SEEGNR HRGCIRINU RETQISS GY REQS INTERNET SEEXRITN HRGTGEGSS,
QSUGRITHDS, QNR THREQTS, E.U. IHSEE, CIRXSES, YIREVQSSS. HENEE, NGX
VISS SEQRN WGTH THEGRETIEQS QSHEETS GY EGDHXTER QNR NETVGRA
SEEXRITN QS VESS QS HGV THQT THEGRN IS QHHSIER IN THE INTERNET. THIS
ANGVSERUE VISS HESH NGX IN RESIUNINU QNR RECESGHINU SEEXRE
QHHSIEQTIGNS QNR NETVGRA HRGTGEGSS QS VESS QS WXISRINU SEEXRE
NETVGRAS.

Going back to letter frequency and a few more guesses!!

Breaking the substitution cipher: example

$\pi =$ $a \mapsto q$ $b \mapsto w$ $c \mapsto e$ $d \mapsto r$ $e \mapsto t$ $f \mapsto y$ $g \mapsto u$ $h \mapsto i$ $i \mapsto o$ $j \mapsto m$ $k \mapsto a$ $l \mapsto s$
$m \mapsto d$ $n \mapsto f$ $o \mapsto g$ $p \mapsto h$ $q \mapsto j$ $r \mapsto k$ $s \mapsto l$ $t \mapsto z$ $u \mapsto x$ $v \mapsto c$ $w \mapsto v$ $x \mapsto b$
$y \mapsto n$ $z \mapsto p$

m = THIS COURSE AIMS TO INTRODUCE YOU TO THE PRINCIPLES AND TECHNIQUES OF SECURING COMPUTERS AND COMPUTER NETWORKS WITH FOCUS ON INTERNET SECURITY. THE COURSE IS EFFECTIVELY SPLIT INTO TWO PARTS. FIRST INTRODUCING THE THEORY OF CRYPTOGRAPHY INCLUDING HOW MANY CLASSICAL AND POPULAR ALGORITHMS WORK E.G. DES, RSA, DIGITAL SIGNATURES, AND SECOND PROVIDING DETAILS OF REAL INTERNET SECURITY PROTOCOLS, ALGORITHMS, AND THREATS, E.G. IPSEC, VIRUSES, FIREWALLS. HENCE, YOU WILL LEARN BOTH THEORETICAL ASPECTS OF COMPUTER AND NETWORK SECURITY AS WELL AS HOW THAT THEORY IS APPLIED IN THE INTERNET. THIS KNOWLEDGE WILL HELP YOU IN DESIGNING AND DEVELOPING SECURE APPLICATIONS AND NETWORK PROTOCOLS AS WELL AS BUILDING SECURE NETWORKS.

Going back to letter frequency and a few more guesses!!

The One-Time Pad (OTP)

- $\mathcal{M} = \mathcal{C} = \mathcal{K} = \{0,1\}^n$
- Encryption: $E(k, m) = k \oplus m$

$$
\begin{array}{rcccccccc}
k & = & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 \\
m & = & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\
\hline
c & = & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0
\end{array}
$$

- Decryption: $D(k, c) = k \oplus c$

$$
\begin{array}{rcccccccc}
k & = & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 \\
c & = & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\
\hline
m & = & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1
\end{array}
$$

- Correctness: $D(k, E(k, m)) = k \oplus (k \oplus m) = m$

Perfect secrecy

Definition

A cipher (E, D) over $(\mathcal{M}, \mathcal{C}, \mathcal{K})$ satisfies perfect secrecy if for all messages $m_1, m_2 \in \mathcal{M}$ and ciphertext $c \in \mathcal{C}$

$$|Pr(E(k, m_1) = c) - Pr(E(k, m_2) = c)| \leq \epsilon$$

where $k \xleftarrow{r} \mathcal{K}$ and ϵ is some "negligible quantity".

OTP satisfies perfect secrecy

Theorem (Shannon 1949)

The One-Time Pad satisfies perfect secrecy

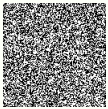
Proof: We first note that for all messages $m \in \mathcal{M}$ and all ciphertexts $c \in \mathcal{C}$

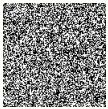
$$\begin{aligned} Pr(E(k,m) = c) &= \frac{\#\{k \in \mathcal{K} \colon k \oplus m = c\}}{\#\mathcal{K}} \\ &= \frac{\#\{k \in \mathcal{K} \colon k = m \oplus c\}}{\#\mathcal{K}} \\ &= \frac{1}{\#\mathcal{K}} \end{aligned}$$

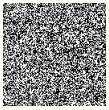
where $k \xleftarrow{r} \mathcal{K}$.
Thus, for all messages $m_1, m_2 \in \mathcal{M}$, and for all ciphertexts $c \in \mathcal{C}$

$$|Pr(E(k,m_1) = c) - Pr(E(k,m_2) = c)| \leq \left| \frac{1}{\#\mathcal{K}} - \frac{1}{\#\mathcal{K}} \right| = 0$$

Two-time pad attacks

m_1 $\oplus$  k =  c_1

 m_2 $\oplus$  k =  c_2

 c_1 $\oplus$  c_2 =  $m_1 \oplus m_2$

Limitations of OTP

- Key-length!
  - The key should be as long as the plaintext
- Getting true randomness!
  - The key should not be guessable from an attacker
  - If the key is not truly random, frequency analysis might again be possible
- Perfect secrecy does not capture all possible attacks
  - OTP is subject to two-time pad attacks
    given $m_1 \oplus k$ and $m_2 \oplus k$, we can compute $m_1 \oplus m_2 = (m_1 \oplus k) \oplus (m_2 \oplus k)$
    English has enough redundancy s.t. $m_1 \oplus m_2 \rightarrow m_1, m_2$
  - OTP is malleable
    given $c = E(k, m)$ with $m = to\ bob \| m_0$, it is possible to compute
    $c' = E(k, m')$ with $m' = to\ eve \| m_0$
    $c' := c \oplus "to\ bob \| 00 \ldots 00" \oplus "to\ eve \| 00 \ldots 00"$

Computer Security

INFR10067

Fall 2025

Cryptography

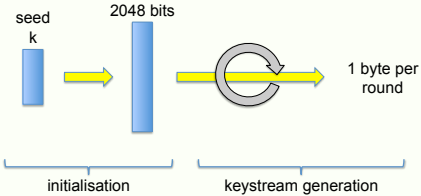
Stream ciphers

Stream ciphers

- Goal: make the OTP practical
- Idea: use a pseudorandom key rather than a really random key
  - The key will not really be random, but will look random
  - The key will be generated from a key seed using a Pseudo-Random Generator (PRG) $G : \{0,1\}^s \to \{0,1\}^n$ with $s << n$
- Encryption using a PRG G: $E(k, m) = G(k) \oplus m$
- Decryption using a PRG G: $D(k, c) = G(k) \oplus c$
- Stream ciphers are subject to two-time pad attacks
- Stream ciphers are malleable

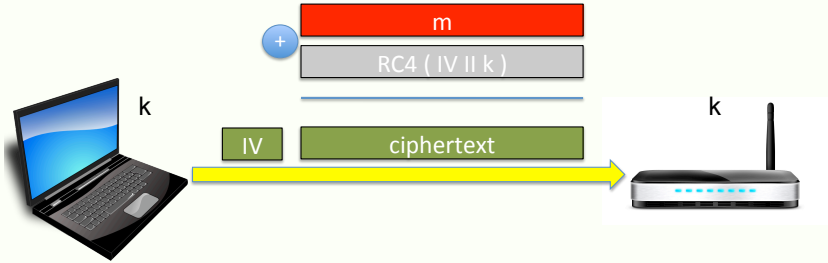
RC4

- Stream cipher invented by Ron Rivest in 1987
- Consists of 2 phases:



- Was used in HTTPS and WEP
- Weaknesses of RC4:
  - first bytes are biased
    ⟶ drop the first 256 generated bytes
  - subject to related keys attacks
    ⟶ choose randomly generated keys as seeds

WEP uses RC4



Initialisation Vector (IV): 24-bits long string

Weaknesses of WEP (Wire Equivalent Privacy for WiFi)

- two-time pad attack: IV is 24 bits long, so the key is reused after at most 2^{24} frames
  $\longrightarrow$ use longer IVs

- Fluhrer, Mantin and Shamir (FMS) attack (related keys attack):
  - the keys only differ in the 24 bits IV
  - first bytes of key stream known because standard headers are always sent
  - for certain IVs knowing m bytes of key and keystream means you can deduce byte $m + 1$ of key

  $\longrightarrow$ instead of using related IVs, generate IVs using a PRG
  $\longrightarrow$ or even better generate message specific seeds using a PRG

Modern stream ciphers

Project eStream: project to “identify new stream ciphers suitable for widespread adoption”, organised by the EU ECRYPT network

⟶ HC-128, Rabbit, Salsa20/12, SOSEMANUK, Grain v1, MICKEY 2.0, Trivium

Conjecture

These eStream stream ciphers are “secure”

Concluding remarks on Stream Ciphers

- Perfect secrecy does not capture all possible attacks.
  $\longrightarrow$ need for different security definition
- Theorem (Shannon 1949) Let (E, D) be a cipher over $(\mathcal{M}, \mathcal{C}, \mathcal{K})$. If (E, D) satisfies perfect secrecy, then the keys should be at least as long as the plaintexts ($|\mathcal{M}| \leq |\mathcal{K}|$).
  $\Rightarrow$ Stream ciphers do not satisfy perfect secrecy because the keys in $\mathcal{K}$ are smaller than the messages in $\mathcal{M}$
  $\longrightarrow$ need for different security definition
- The design of crypto primitives is subtle and error prone.
  $\longrightarrow$ use standardised publicly know primitives
- Crypto primitives are secure under a precisely defined threat model.
  $\longrightarrow$ respect the security assumptions of the crypto primitives
- Many attacks due to poor implementations of cryptography

https://blog.cryptographyengineering.com/2025/09/10/kerberoasting/

Kerberoasting — one-slide summary

Offline password theft: attacker extracts Kerberos service tickets and performs **bruteforce** offline:

- Ticket is encrypted with a key derived from the service account **password**.
- Legacy ciphers (e.g. **RC4**) and NT-hash modes make cracking far faster.
- Real-world risk: human-chosen service passwords remain exploitable.
- Mitigations: use autogenerated long keys, disable RC4, enforce rotation and strong passwords.

Real-world impact: Kerberoasting remains a live threat (linked to high-impact incidents such as the May 2024 Ascension Health ransomware attack) because legacy options are still enabled and admins often use human-chosen service passwords.

THE UNIVERSITY of EDINBURGH

Computer Security

INFR10067

Fall 2025

Cryptography

Block Ciphers

Recap

Action items:

- Suggest exercises for practicing the use of notation.
- Hard to introduce every symbol, e.g. $\|$, $\#\{\cdot\}$, $|\{\cdot\}|$,
- ✓ Keep on interrupting me, when I lose you!
- ✓ Request: Practical examples from case studies and news.

We made it until Block Ciphers section.

Block ciphers

A block cipher with parameters k and ℓ is a pair of deterministic algorithms (E, D) such that

- Encryption $E : \{0,1\}^k \times \{0,1\}^\ell \to \{0,1\}^\ell$
- Decryption $D : \{0,1\}^k \times \{0,1\}^\ell \to \{0,1\}^\ell$

Examples:

3DES: $\ell = 64, k = 168$

AES: $\ell = 128, k = 128, 192, 256$

Notation:

$\mathcal{K} = \{0,1\}^k, \mathcal{M} = \mathcal{C} = \{0,1\}^\ell$.

We use capital letters for blocks, i.e. $K \in \mathcal{K}, M \in \mathcal{M}, C \in \mathcal{C}$.

Data Encryption Standard (DES)

- Early 1970s: Horst Feistel designs Lucifer at IBM
  $k = 128$ bits, $\ell = 128$ bits
- 1973: NBS calls for block cipher proposals.
  $\longrightarrow$ IBM submits a variant of Lucifer.
- 1976: NBS adopts DES as a federal standard
  $k = 56$ bits, $\ell = 64$ bits
- 1997: DES broken by exhaustive search
- 2001: NIST adopts AES to replace DES
  $k = 128, 192, 256$ bits, $\ell = 128$ bits

Was widely deployed in banking (ATM machines) and commerce - now deprecated

Attacks on DES

- **Exhaustive search:** it takes 2^{56} to do an exhaustive search over the key space
  $\longrightarrow$ COBACOBANA (120 FPGAs, $\sim$ 10K$): 7 days

- **Linear cryptanalysis:** found affine approximations to DES
  $\longrightarrow$ can find 14 key bits in time 2^{42}
  brute force the remaining 56-14=42 in time 2^{42}
  $\Rightarrow$ total attack time $\approx 2^{43}$

$\Rightarrow$ DES is badly broken! Do not use it in new projects!!

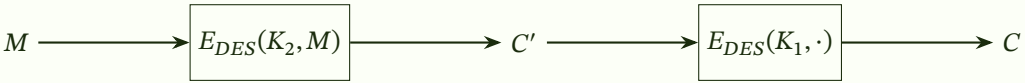
Triple DES (3DES)

- Goal: build on top of DES a block cipher resistant against exhaustive search attacks
- Used in bank cards and RFID chips
- Let $DES = (E_{DES}, D_{DES})$. We build $3DES = (E_{3DES}, D_{3DES})$ as follows
  - $E_{3DES} : (\{0,1\}^k)^3 \times \{0,1\}^\ell \to \{0,1\}^\ell$
    $E_{3DES}((K_1, K_2, K_3), M) = E_{DES}(K_1, D_{DES}(K_2, E_{DES}(K_3, M))) \longrightarrow K_1 = K_2 = K_3 \Rightarrow DES$
  - $D_{3DES} : (\{0,1\}^k)^3 \times \{0,1\}^\ell \to \{0,1\}^\ell$
    $D_{3DES}((K_1, K_2, K_3), C) = D_{DES}(K_3, E_{DES}(K_2, D_{DES}(K_1, C)))$

  $\longrightarrow$ 3 times as slow as DES!!
- key-size = $3 \times 56 = 168$ bits
  $\Rightarrow$ Exhaustive search attack in 2^{168}
- simple (meet-in-the-middle) attack in time 2^{118}

What about double DES (2DES)?

- $E_{2DES}((K_1, K_2), M) = E_{DES}(K_1, E_{DES}(K_2, M))$



For M and C such that $E_{2DES}((K_1, K_2), M) = C$ we have that

$$E_{DES}(K_2, M) = C' = D_{DES}(K_1, C)$$

- Meet-in-the-middle attack reduces the time for key recovery from 2^{112} to 2^{63}.
  Given $\mathbf{M} = (M_1, \ldots, M_{10})$ and $\mathbf{C} = (E_{2DES}((K_1, K_2), M_1), \ldots, E_{2DES}((K_1, K_2), M_{10}))$

  - For all possible K_2, *compute* $E_{DES}(K_2, \mathbf{M})$
  - Sort table according to the resulting $E_{DES}(K_2, \mathbf{M})$

  } $2^{56}\log(2^{56})$

  - For each possible K_1, *compute* $D_{DES}(K_1, \mathbf{C})$
  - Look up in the table if $D_{DES}(K_1, C) = E_{DES}(K_2, \mathbf{M})$

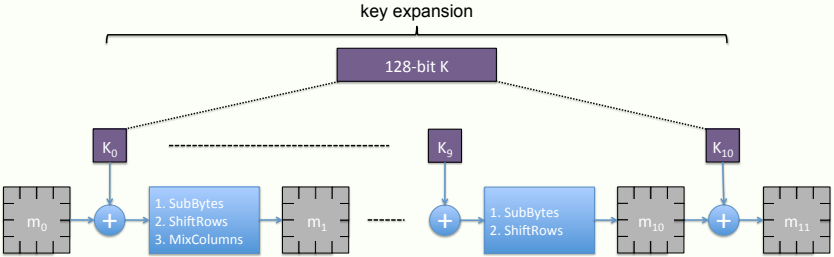
  } $2^{56}\log(2^{56})$

  $\Rightarrow$ time $< 2^{63}$

The Advanced Encryption Standard (AES)

- Goal: replace 3DES which is too slow (3DES is 3 times as slow as DES)
- 2001: NIST adopts Rijndael as AES
- Block size $\ell = 128$ bits, Key size $k = 128, 192, 256$ bits

AES: encryption circuit



- m_i : 4×4 byte matrix, K_i: 128-bit key
- m_0: plaintext M, m_{11}: ciphertext C
- at the last round MixColumns is not applied

Attacks on AES

- **Related-key attack** on the 192-bit and 256-bit versions of AES: exploits the AES key schedule [A. Biryukov, D. Khovratovich (2009)]
  $\longrightarrow$ key recovery in time $\sim 2^{99}$

- First **key-recovery attack** on full AES [A. Bogdanov, D. Khovratovich, C. Rechberger (2011)]
  $\longrightarrow$ 4 times faster than exhaustive search

- Even quantum computers offer only modest advantage (Grover's algorithm reduces security to 2^{64} operations).
  $\longrightarrow$ More on quantum computers and how they affect cryptography later in the course.

$\Rightarrow$ Existing attacks on AES-128 are still not practical, but should use AES-192 and AES-256 in new projects!

Computer Security

INFR10067

Fall 2025

Cryptography

Using block ciphers

Goal

Encrypt M using a block cipher operating on blocks of length ℓ when $|M| \neq \ell$

Padding - $|M| \leq \ell$

- Bit padding - append a *set bit* ('1') at the end of message, and then append as many *reset bits* ('0') required.

  Example: padding a 52-bits message for a 64-bits block:

  `11010011 01010110 10010000 00111010 10110101 01011010 11111000 00000000`

  Padding a 64-bits message M for 64-bits blocks requires adding a padding block:

  `M‖10000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000`

- ANSI X.923 - byte padding - pad with zeros, the last byte defines the number of padded bytes.

  Example: padding a 4-bytes message for 8-bytes blocks:

  `DD DD DD DD 00 00 00 04`

  Padding a $8k$-bytes messages for 8-bytes blocks requires adding a padding block:

  `DD DD DD DD DD DD DD DD ‖ 00 00 00 00 00 00 00 08`

- PKCS#7 - byte padding - the value of each added byte is the total number of padding bytes. The padding will be 01, or 02 02, or 03 03 03, or 04 04 04 04, *etc.*

  Example: padding a 4-bytes message for 8-bytes blocks:

  `DD DD DD DD 04 04 04 04`

  Padding a 8-bytes message for 8-bytes blocks requires adding a padding block:

  `DD DD DD DD DD DD DD DD ‖ 08 08 08 08 08 08 08 08`

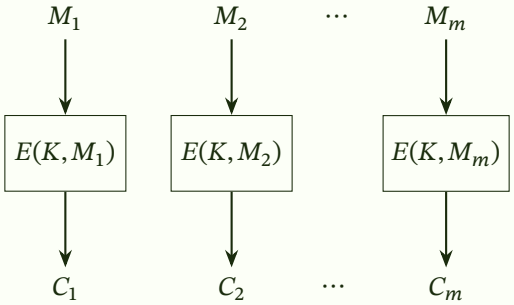
Electronic Code Book (ECB) mode

(E, D) a block cipher.

To encrypt message M under key K using ECB mode:

- M is padded:
  $\Rightarrow\ M' = M||P$ such that $|M'| = m \times \ell$ for some m.
- M' is broken into m blocks of length ℓ
  $\Rightarrow\ M' = M_1 \,||\, M_2 \,||\, \ldots \,||\, M_m$
- Each block M_i is encrypted under the key K using the block cipher
  $\Rightarrow\ C_i = E(K, M_i)$ for all $i \in \{1, \ldots, m\}$
- The ciphertext corresponding to M is the concatenation of the C_is
  $\Rightarrow\ C = C_1 \,||\, C_2 \,||\, \ldots \,||\, C_m$

Weakness of ECB



Problem: $\forall i, j.\, M_i = M_j \Rightarrow c_i = E(k, M_i) = E(k, M_j) = c_j$

$\Rightarrow$ Malleable and vulnerable agains frequency analysis!

Weakness of ECB in pictures

Original image

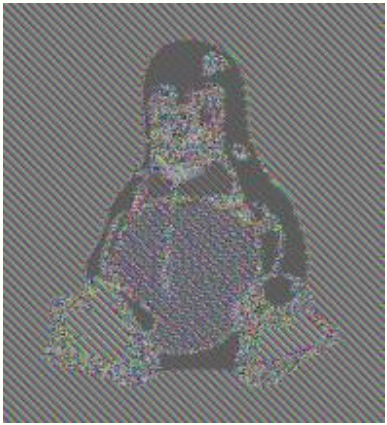
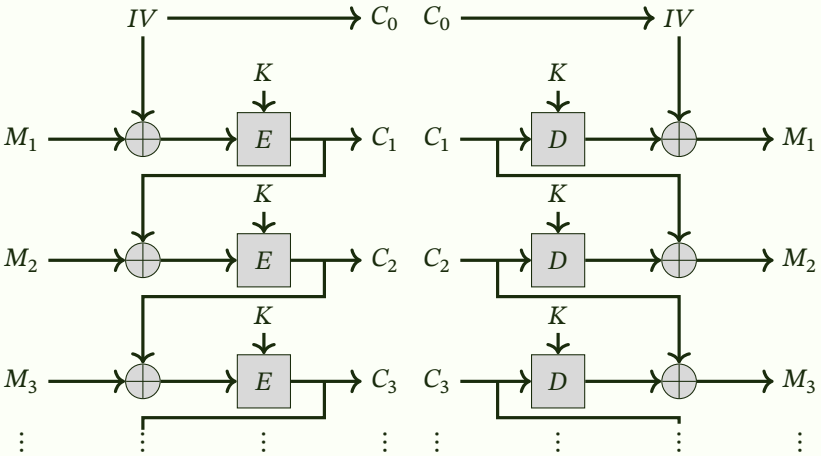


Image encrypted using ECB mode

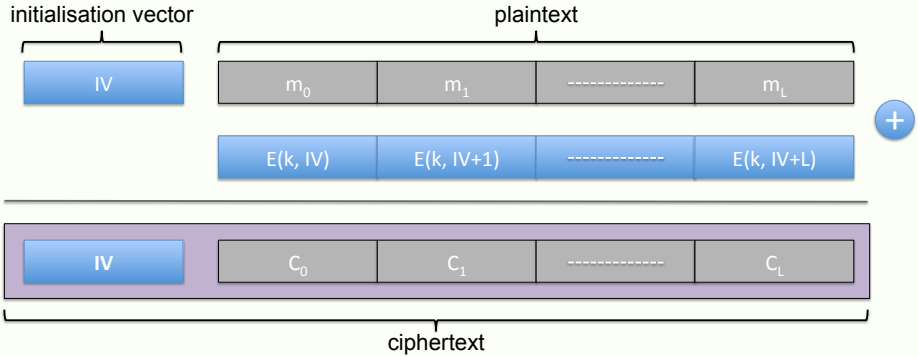
Cipher-block chaining (CBC) mode

(E, D) a block cipher. IV chosen at random in $\{0,1\}^{\ell}$.



Counter (CTR) mode

(E, D) a block cipher that manipulates blocks of size ℓ.



IV chosen at random in $\{0,1\}^{\ell}$

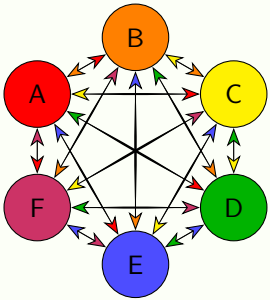
Block-size is also a problem!

- Sweet32 - birthday attacks on 64-bit block ciphers in TLS and openVPN
- Attack due to block-size being too small

The key management problem

The confidentiality problem is now reduced to a key management problem:

- Where are keys generated?
- How are keys generated?
- How are keys shared?
- Where are keys stored?
- Where are the keys actually used?
- How are key revoked and replaced?



One shared secret key per pair of users that want to communicate

What we have learned on Symmetric Encryption

- Frequency analysis as a cryptanalysis attack on classic encryption
- Importance of randomness in cryptography
- Stream ciphers
  - simple and efficient symmetric encryption schemes
  - use a random IV to thwart two-time pad attacks
  - subject to malleability attacks
- Block ciphers - use AES not DES
- CBC mode is more secure that ECB but less resilient to packets loss
- CTR mode more secure than ECB and parallelisable
- Keep up to date with cryptanalytic advances and standards.
  Modern symmetric encryption also guarantees authenticity
  → no malleability
- Do not implement crypto lightly - use public reference implementations

Authenticated Encryption: formal definition

Authenticated Encryption

A SKE scheme Σ is a secure authenticated encryption (AE) scheme if the following two libraries are indistinguishable:

$\mathcal{L}^{\Sigma}_{\text{ae-real}}$

$K \twoheadleftarrow \Sigma.\mathcal{K}$

$\underline{\text{AE.ENC}(M)}$:
return $\Sigma.\text{Enc}(K, M)$

$\underline{\text{AE.DEC}(C)}$:
return $\Sigma.\text{Dec}(K, C)$

$\approx$

$\mathcal{L}^{\Sigma}_{\text{ae-rand}}$

$\underline{\text{AE.ENC}(M)}$:
$C \twoheadleftarrow \Sigma.\mathcal{C}(|M|)$
$\mathcal{D}[C] \coloneqq M$
return C

$\underline{\text{AE.DEC}(C)}$:
if $\mathcal{D}[C]$ defined: return $\mathcal{D}[C]$
else: return err

THE UNIVERSITY of EDINBURGH

Computer Security

INFR10067

Fall 2025

Cryptography

Cryptographic hash functions and MACs

Markulf Kohlweiss
School of Informatics
University of Edinburgh

Introduction

Encryption $\Rightarrow$ confidentiality against eavesdropping

What about authenticity and integrity against an active attacker?
$\longrightarrow$ cryptographic hash functions and Message authentication codes
$\longrightarrow$ this lecture

Building block for constructing authenticated encryption schemes, bitcoin proof of work, Merkle trees, protecting passwords, and much more.

One-way functions (OWFs)

A OWF is a function that is easy to compute but hard to invert:

Definition (One-way)

A function f is a one-way function if for all y there is no efficient algorithm which can compute x such that $f(x) = y$

Constant functions ARE NOT OWFs
- any x is such that $f(x) = c$

The successor function in $\mathbb{N}$ IS NOT a OWF
- given $succ(n)$ it is easy to retrieve $n = succ(n) - 1$

Multiplication of large primes IS a OWF:
- integer factorisation is a hard problem - given $p \times q$ (where p and q are primes) it is hard to retrieve p and q

Collision-resistant functions (CRFs)

A function is a CRF if it is hard to find two messages that get mapped to the same value threw this function

Definition (Collision resistance)

A function f is collision resistant if there is no efficient algorithm that can find two messages m_1 and m_2 such that $f(m_1) = f(m_2)$

Constant functions ARE NOT CRFs
for all m_1 and m_2, $f(m_1) = f(m_2)$

The successor function in $\mathbb{N}$ IS a CRF
the predecessor of a positive integer is unique

Multiplication of large primes IS a CRF:
every positive integer has a unique prime factorisation

Cryptographic hash functions

A cryptographic hash function takes messages of arbitrary length and returns a fixed-size bit string such that any change to the data will (with very high probability) change the corresponding hash value.

Definition (Cryptographic hash function)

A cryptographic hash function $H : \mathcal{M} \to \mathcal{T}$ is a function that satisfies the following 4 properties:

- $|\mathcal{M}| >> |\mathcal{T}|$
- it is easy to compute the hash value for any given message
- it is hard to retrieve a message from it hashed value (OWF)
- it is hard to find two different messages with the same hash value (CRF)

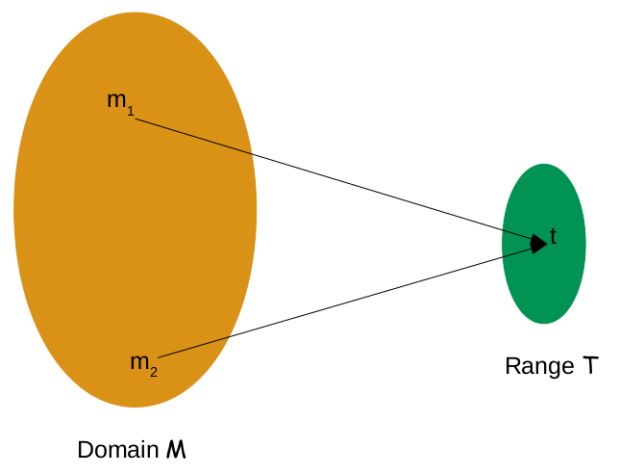
Examples: MD4, MD5, SHA-1, RIPEMD160, SHA-256, SHA-512, SHA-3...

⇝In new projects use SHA-256 or SHA-512 or SHA-3

Cryptographic hash functions: applications

- **Commitments -** Allow a participant to commit to a value v by publishing the hash $H(v)$ of this value, but revealing v only later. Ex: electronic voting protocols, digital signatures, ...
- **File integrity -** Hashes are sometimes posted along with files on "read-only" spaces to allow verification of integrity of the files. Ex: SHA-256 is used to authenticate Debian GNU/Linux software packages
- **Password verification -** Instead of storing passwords in cleartext, only the hash digest of each password is stored. To authenticate a user, the password presented by the user is hashed and compared with the stored hash.
- **Key derivation -** Derive new keys or passwords from a single, secure key or password.
- **Building block of other crypto primitives -** Used to build MACs, block ciphers, PRG, ...

Collisions are unavoidable



The domain being much larger than the range, collisions necessarily exist

The birthday attack - attack on all schemes

Theorem

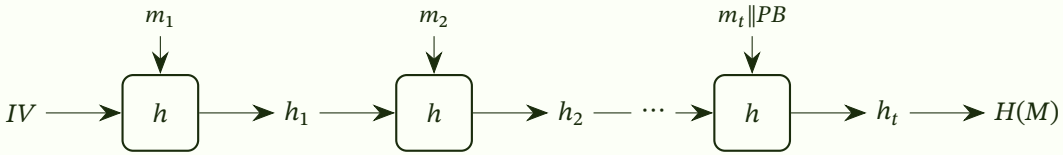
Let $H : \mathcal{M} \to \{0,1\}^n$ be a cryptographic hash function ($|\mathcal{M}| >> 2^n$)
Generic algorithm to find a collision in time $O(2^{n/2})$ hashes:

1. *Choose $2^{n/2}$ random messages in $\mathcal{M}$: $m_1, \dots, m_{2^{n/2}}$*
2. *For $i = 1, \dots, 2^{n/2}$ compute $t_i = H(m_i)$*
3. *If there exists a collision ($\exists i, j.\ t_i = t_j$)*
   then return (m_i, m_j)
   else go back to 1

Birthday paradox Let $r_1, \dots, r_\ell \in \{1, \dots, N\}$ be independent variables. For $\ell = 1.2 \times \sqrt{N}$,
$Pr(\exists i \neq j.\ r_i = r_j) \geq \frac{1}{2}$

$\Rightarrow$ the expected number of iteration is 2
$\Rightarrow$ running time $O(2^{n/2})$

The Merkle-Damgard construction



Merkle–Damgård construction

- Compression function: $h : \mathcal{T} \times \mathcal{X} \to \mathcal{T}$
- PB: $1000 \ldots 0$||mes-len (add extra block if needed) (different variants!)

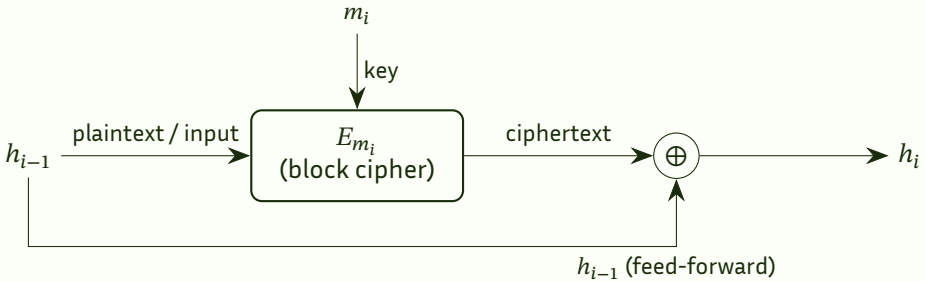
Theorem

Let H be built using the MD construction to the compression function h. If H admits a collision, so does h.

Example of MD constructions: MD5, SHA-1, SHA-2, ...

Compression functions from block ciphers

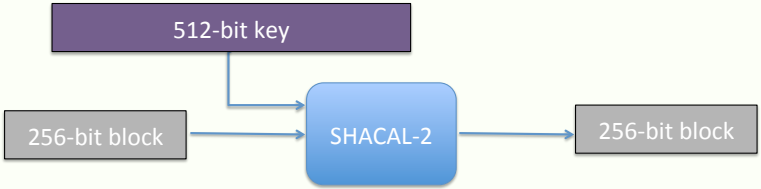
Let $E : \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ be a block cipher



Davies–Meyer: $h_i = E_{m_i}(h_{i-1}) \oplus h_{i-1}$

Example of cryptographic hash function: SHA-256

- Structure: Merkle-Damgard
- Compression function: Davies-Meyer
- Bloc cipher: SHACAL-2



THE UNIVERSITY of EDINBURGH

Computer Security

INFR10067

Fall 2025

Cryptography

Message Authentication Codes (MACs)

Encryption is not always enough

$e = E(K_E, \text{Transfer 100 € on Bob's account})$ →

Royal Bank of Scotland

What if the encryption scheme E is the OTP - $e = K_E \oplus$ Transfer 100 € on Bob's account?

e →

$e \oplus 0\ldots0\text{Bob}0\ldots0 \oplus 0\ldots0\text{Eve}0\ldots0$ →
$= E(K_E, \text{Transfer 100 € on Eve's account})$

Royal Bank of Scotland

Goal: message integrity

A MAC is a pair of algorithms (S, V) defined over $(\mathcal{K}, \mathcal{M}, \mathcal{T})$:

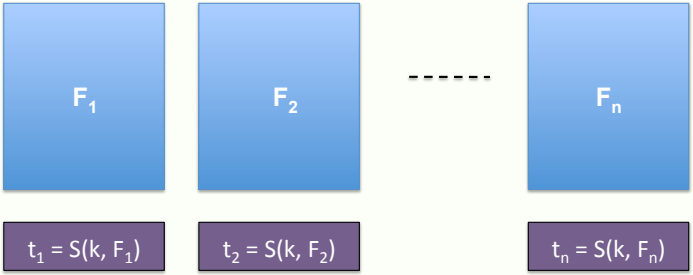
- $S : \mathcal{K} \times \mathcal{M} \to \mathcal{T}$
- $V : \mathcal{K} \times \mathcal{M} \times \mathcal{T} \to \{\top, \bot\}$
- Consistency: $V(k, m, S(k, m)) = T$

Unforgeability

It is hard to computer a valid pair $(m, S(k, m))$ without knowing k

File system protection

- At installation time



k derived from user password

- To check for virus file tampering/alteration:
  - reboot to clean OS
  - supply password
  - any file modification will be detected

Block ciphers and message integrity

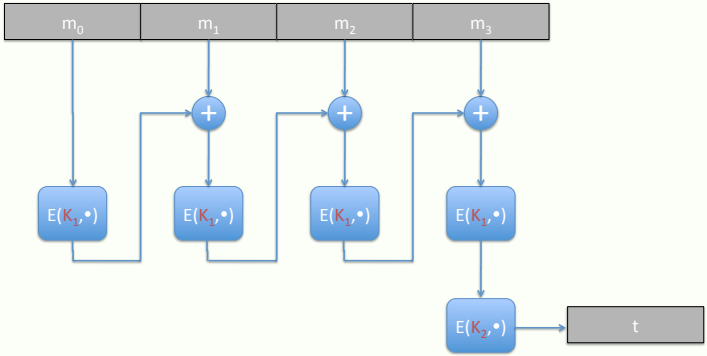
Let (E, D) be a block cipher. We build a MAC (S, V) using (E, D) as follows:

- $S(k, m) = E(k, m)$
- $V(k, m, t) =$ if $m = D(k, t)$
  then return $\top$
  else return $\bot$

But: block ciphers can usually process only 128 or 256 bits

Our goal now: construct MACs for long messages

ECBC-MAC

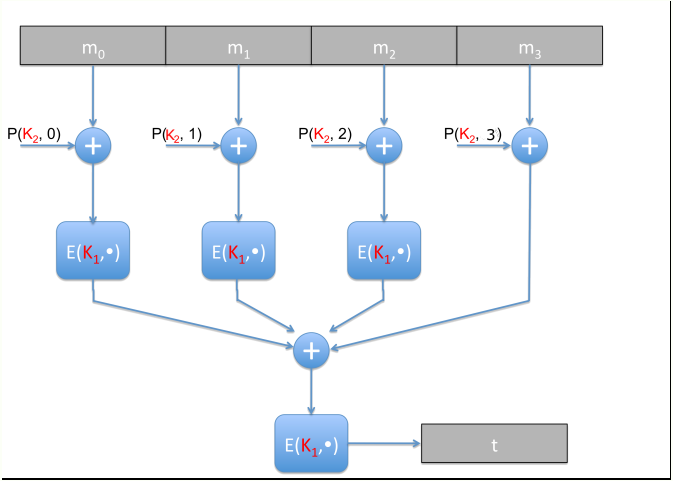


- $E : \mathcal{K} \times \{0,1\}^n \rightarrow \{0,1\}^n$ a block cipher
- $ECBC\text{-}MAC : \mathcal{K}^2 \times \{0,1\}^* \rightarrow \{0,1\}^n$

→ the last encryption is crucial to avoid forgeries!!

Ex: 802.11i uses AES based ECBC-MAC

PMAC



- $E : \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ a block cipher
- $P : \mathcal{K} \times \mathbb{N} \to \{0,1\}^n$ an easy to compute function
- $PMAC : \mathcal{K}^2 \times \{0,1\}^* \to \{0,1\}^n$

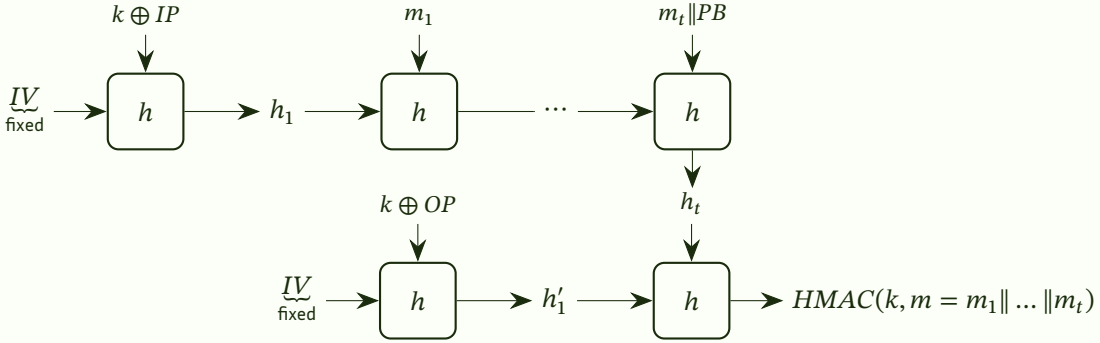
See also: https://www.youtube.com/watch?v=gZiBYDX9Fpo

HMAC

MAC built from cryptographic hash functions

$$HMAC(k, m) = H(k \oplus OP || H(k \oplus IP || m))$$

IP, OP: publicly known padding constants



HMAC construction

Naive MAC from Hashing

Simplest way to build a MAC from a hash function, prepend the key:

$$MAC(k, m) = H(k \| m)$$

This is not generally secure, but works for SHA3/Keccak as it prevents length extension attack.

Source: `https://keccak.team/keccak_strengths.html`

See also: `https://crypto.stackexchange.com/questions/1070/why-is-hk-mathbin-vert-x-not-a-secure-mac-construction`

THE UNIVERSITY of EDINBURGH

Computer Security

INFR10067

Fall 2025

Cryptography

Authenticated encryption

Plain encryption is malleable

- The decryption algorithm never fails
- Changing one bit of the i^{th} block of the ciphertext
  - CBC decryption: will affect last blocks after the i^{th} of the plaintext
  - ECB decryption: will only the i^{th} block of the plaintext
  - CTR decryption: will only affect one bit of the i^{th} block of the plaintext

Decryption should fail if a ciphertext was not computed using the key

Goal

Simultaneously provide data confidentiality, integrity and authenticity
⇝ decryption combined with integrity verification in one step

Encrypt-then-MAC

1. Always compute the MACs on the ciphertext, never on the plaintext
2. Use two different keys, one for encryption (K_E) and one for the MAC (K_M)

Encryption

1. $C \leftarrow E_{AES}(K_E, M)$
2. $T \leftarrow HMAC\text{-}SHA(K_M, C)$
3. return $C||T$

Decryption

1. if $T = HMAC\text{-}SHA(K_M, C)$
2. then return $D_{AES}(K_E, C)$
3. else return $\perp$

Do not:

- Encrypt-and-MAC: $E_{AES}(K_E, M)||HMAC\text{-}SHA(K_M, M)$
- MAC-then-Encrypt: $E_{AES}(K_E, M||HMAC\text{-}SHA(K_M, M))$

AES GCM

Galois Counter Mode

Combines

1. Galois field based One-time MAC for authentication
2. AES based Counter Mode for encryption

- Trick: One-time MAC is encrypted too
  $\Rightarrow$ secure for many messages
- Widely adopted for its performance
- Many good implementations of this mode

Computer Security

INFR10067

Fall 2025

Cryptography

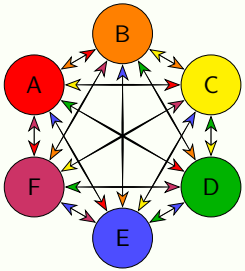
Asymmetric encryption

Markulf Kohlweiss
School of Informatics
University of Edinburgh

Introduction

So far: how two users can protect data using a shared secret key

- One shared secret key per pair of users that want to communicate



Our goal now: how to establish a shared secret key to begin with?

- Trusted Third Party (TTP)
- Diffie-Hellman (DH) protocol
- RSA
- ElGamal (EG)

Recap

Action items:

- ✓ Upload slides in advance (note that they might change before, and even after the lecture)
- ✓ Exercises: Those that struggle. please read this short chapter and do the exercises there: `https://joyofcryptography.com/pdf/chap0.pdf`
- ✓ **Office hours** are 11:15 Mon and Fri. Talk with me after lecture, especially MSc students. Offer to discuss Feistel networks, S-boxes, and key schedules stands.
- ✓ Request: What are challenges in modern cryptography?
  Trusted, distributed, fair and safe AI. **Digital and Decentralized Identity.** Post-quantum. Formalization of cryptography in theorem provers, e.g. Lean.

We finished MAC algorithms and authenticated encryption last time.

Online Trusted Third Party (TTP)

- Users U_1, U_2, U_3, ..., U_n, ...
- Each user U_i has a shared secret key K_i with the TTP
- U_i and U_j can establish a key $K_{i,j}$ with the help of the TTP
- $\{m\}_k$ denotes the symmetric encryption of m under the key k

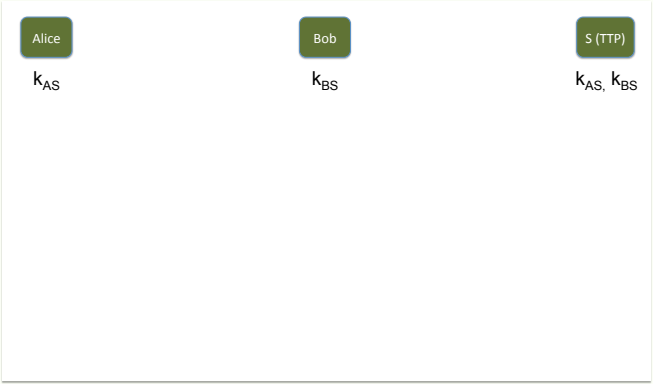


Figure: Paulson's variant of the Yahalom protocol

Online Trusted Third Party (TTP)

- Users U_1, U_2, U_3, ..., U_n, ...
- Each user U_i has a shared secret key K_i with the TTP
- U_i and U_j can establish a key $K_{i,j}$ with the help of the TTP
- $\{m\}_k$ denotes the symmetric encryption of m under the key k

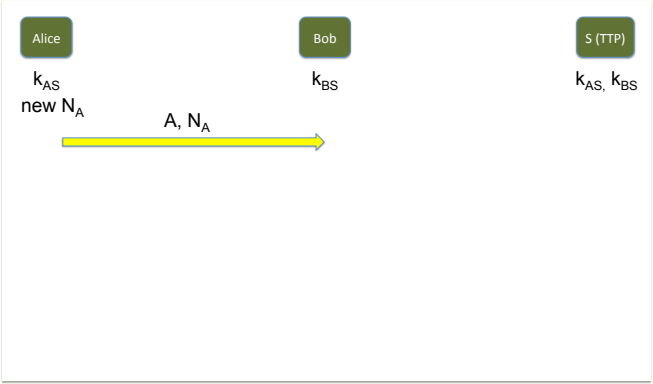


Figure: Paulson's variant of the Yahalom protocol

Online Trusted Third Party (TTP)

- Users U_1, U_2, U_3, ..., U_n, ...
- Each user U_i has a shared secret key K_i with the TTP
- U_i and U_j can establish a key $K_{i,j}$ with the help of the TTP
- $\{m\}_k$ denotes the symmetric encryption of m under the key k

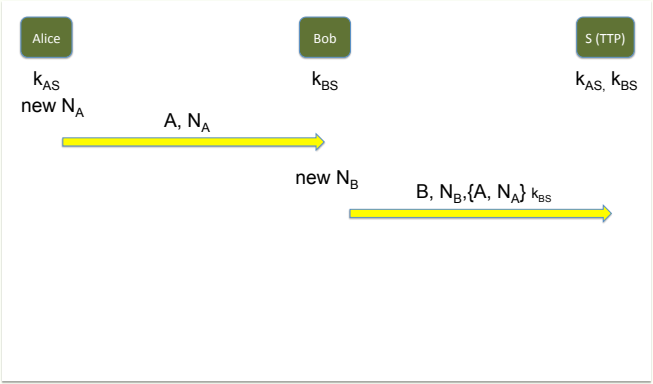


Figure: Paulson's variant of the Yahalom protocol

Online Trusted Third Party (TTP)

- Users U_1, U_2, U_3, ..., U_n, ...
- Each user U_i has a shared secret key K_i with the TTP
- U_i and U_j can establish a key $K_{i,j}$ with the help of the TTP
- $\{m\}_k$ denotes the symmetric encryption of m under the key k

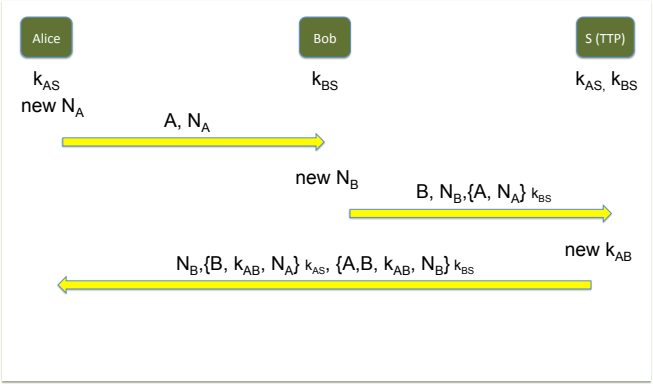


Figure: Paulson's variant of the Yahalom protocol

Online Trusted Third Party (TTP)

- Users U_1, U_2, U_3, ..., U_n, ...
- Each user U_i has a shared secret key K_i with the TTP
- U_i and U_j can establish a key $K_{i,j}$ with the help of the TTP
- $\{m\}_k$ denotes the symmetric encryption of m under the key k

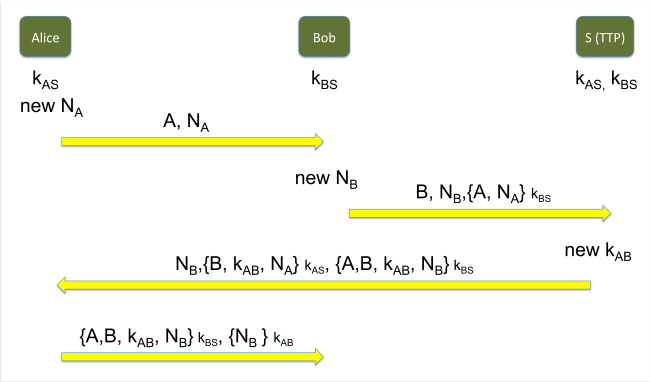


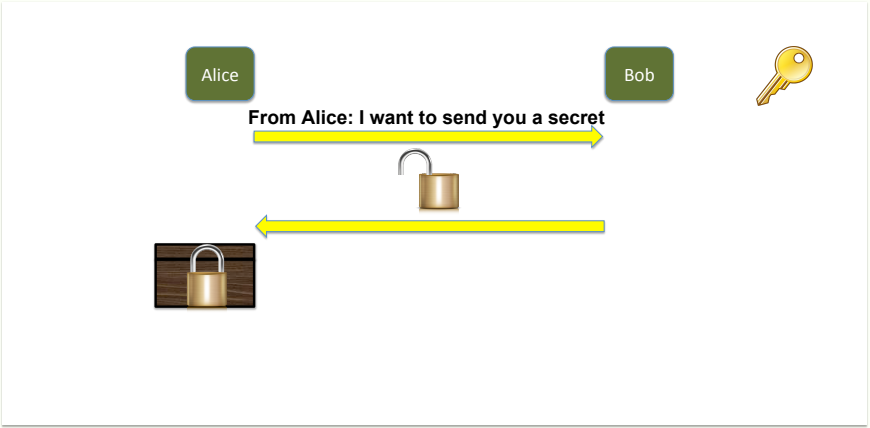
Figure: Paulson's variant of the Yahalom protocol

Goal of public-key encryption

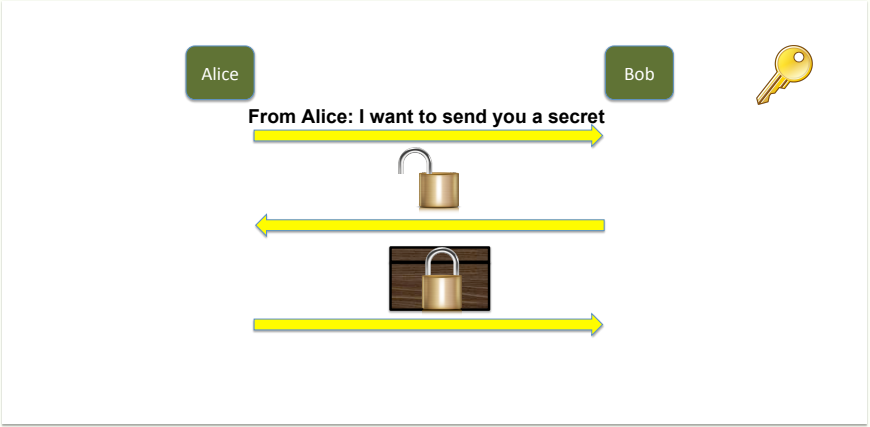
Goal of public-key encryption

Goal of public-key encryption

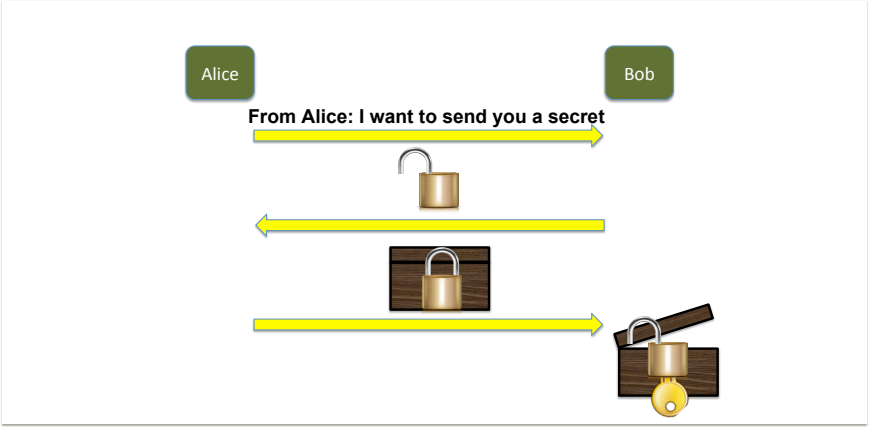
Goal of public-key encryption



Goal of public-key encryption



Goal of public-key encryption



Public-key encryption - Definition

- key generation algorithm: $G :\rightarrow \mathcal{K}_{pk} \times \mathcal{K}_{sk}$
  encryption algorithm $E : \mathcal{K}_{pk} \times \mathcal{M} \rightarrow \mathcal{C}$
  decryption algorithm $D : \mathcal{K}_{sk} \times \mathcal{C} \rightarrow \mathcal{M}$
  st. $\forall (sk, pk) \in G$, and $\forall m \in \mathcal{M}, D(sk, E(pk, m)) = m$

- the decryption key sk_{Bob} is secret (only known to Bob). The encryption key pk_{Bob} is known to everyone. And $sk_{Bob} \neq pk_{Bob}$

THE UNIVERSITY of EDINBURGH

Computer Security

INFR10067

Fall 2025

Asymmetric encryption

We need a bit of number theory now

Primes

Definition

$p \in \mathbb{N}$ is a **prime** if its only divisors are 1 and p

Ex: 2, 3, 5, 7, 11, 13, 17, 19, 23, 29

Theorem

Every $n \in \mathbb{N}$ has a **unique factorization** *as a product of prime numbers (which are called its factors)*

Ex: $23244 = 2 \times 2 \times 3 \times 13 \times 149$

Relative primes

Definition

a and b in $\mathbb{Z}$ are **relative primes** if they have no common factors

Definition

The Euler function $\phi(n)$ is the number of elements that are relative primes with n:

$$\phi(n) = |\{m \mid 0 < m < n \text{ and } \gcd(m, n) = 1\}|$$

- For p prime: $\phi(p) = p\text{-}1$
- For p and q primes: $\phi(p \cdot q) = (p\text{-}1)(q\text{-}1)$

Integers modulo n: $\mathbb{Z}_n$

- Let $n \in \mathbb{N}$. We define $\mathbb{Z}_n = \{0, \dots n\text{-}1\}$

$$\forall a \in \mathbb{Z}_n,\ \forall b \in \mathbb{Z},\ a \equiv b \pmod{n} \ \Leftrightarrow\ \exists k \in \mathbb{Z}.\ b = a + k \cdot n$$

- Modular inversion: the inverse of $x \in \mathbb{Z}_n$ is $y \in \mathbb{Z}_n$ *s.t.* $x \cdot y \equiv 1 \pmod{n}$. We denote x^{-1} the inverse of x mod n
  Ex: 7^{-1} in $\mathbb{Z}_{12}$: 7
  4^{-1} in $\mathbb{Z}_{12}$: 4 has no inverse in $\mathbb{Z}_{12}$

Theorem

Let $n \in \mathbb{N}$. Let $x \in \mathbb{Z}_n$. x has a inverse in $\mathbb{Z}_n$ iff $\gcd(x, n) = 1$

Inverse is computed using the Extended Euclidean Algorithm.

Multiplicative group of integers modulo n: $\mathbb{Z}_n^*$

- Let $n \in \mathbb{N}$. We define $\mathbb{Z}_n^* = \{x \in \mathbb{Z}_n \mid \gcd(x, n) = 1\}$
  Ex: $\mathbb{Z}_{12} = \{1, 5, 7, 11\}$
- Note that $|\mathbb{Z}_n^*| = \phi(n)$

Theorem (Euler)

$\forall n \in \mathbb{N}, \forall x \in \mathbb{Z}_n^*, x^{\phi(n)} \equiv 1 \pmod{n}$

Theorem (Euler)

$\forall p$ prime, $\mathbb{Z}_p^*$ is a cyclic group, i.e.

$$\exists g \in \mathbb{Z}_p^*, \{1, g, g^2, g^3, \dots, g^{p-2}\} = \mathbb{Z}_p^*$$

Intractable problems

- Factoring:
  input: $n \in \mathbb{N}$
  output: $p_1, \ldots, p_m$ primes *st.* $n = p_1 \cdot \cdots \cdot p_m$
- RSA Problem
  input: n *st.* $n = p \cdot q$ with $2 \leq p, q$ primes
  e *st.* $\gcd(e, \phi(n)) = 1$
  $m^e \mod n$
  output: m
- Discrete Logarithm Problem (DLP):
  input: prime p, generator g of $\mathbb{Z}_p^*$, $y \in \mathbb{Z}_p^*$
  output: x such that $y = g^x \pmod p$
- Diffie-Hellman Problem (DHP):
  input: prime p, generator g of $\mathbb{Z}_p^*$, $g^a \pmod p$, $g^b \pmod p$
  output: $g^{ab} \pmod p$

THE UNIVERSITY of EDINBURGH

Computer Security

INFR10067

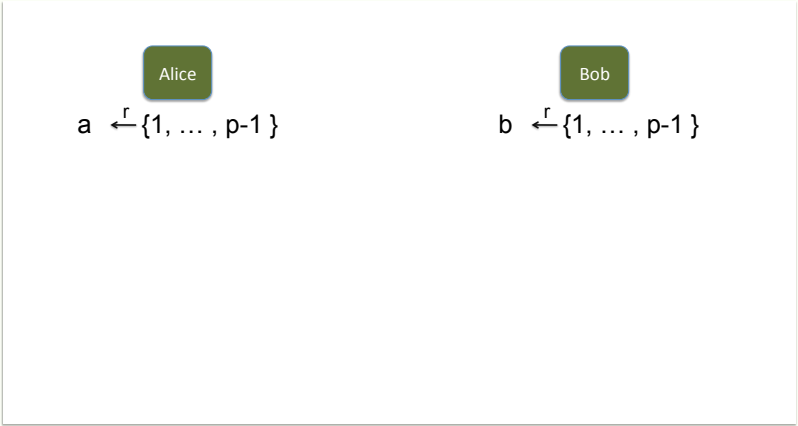
Fall 2025

Asymmetric encryption

Establish a key without a TTP

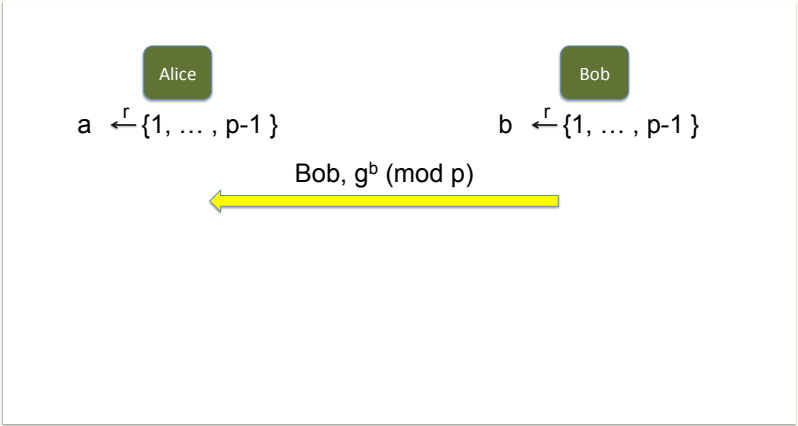
The Diffie-Hellman (DH) protocol

- Assumption: the DHP is hard in $\mathbb{Z}_p^*$
- Fix a very large prime p, and g generator of $\mathbb{Z}_p^*$



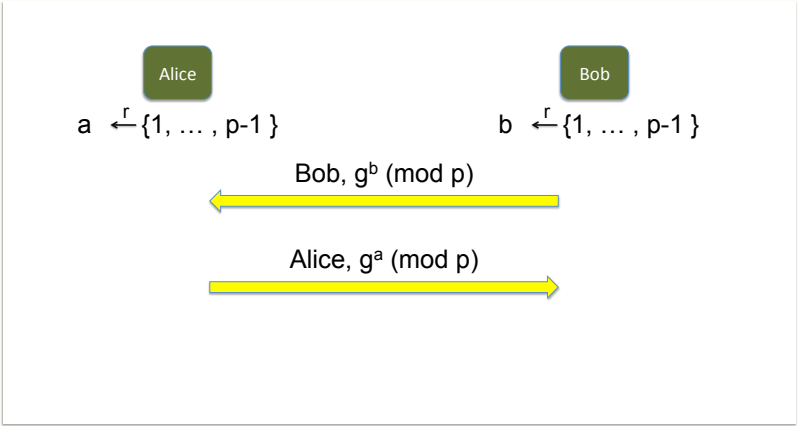
The Diffie-Hellman (DH) protocol

- Assumption: the DHP is hard in $\mathbb{Z}_p^*$
- Fix a very large prime p, and $g \in \{1, \dots, p\text{-}1\}$



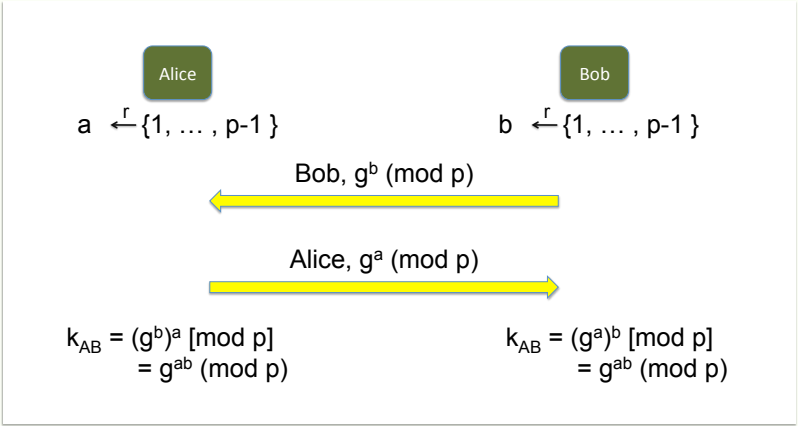
The Diffie-Hellman (DH) protocol

- Assumption: the DHP is hard in $\mathbb{Z}_p^*$
- Fix a very large prime p, and $g \in \{1, \dots, p\text{-}1\}$

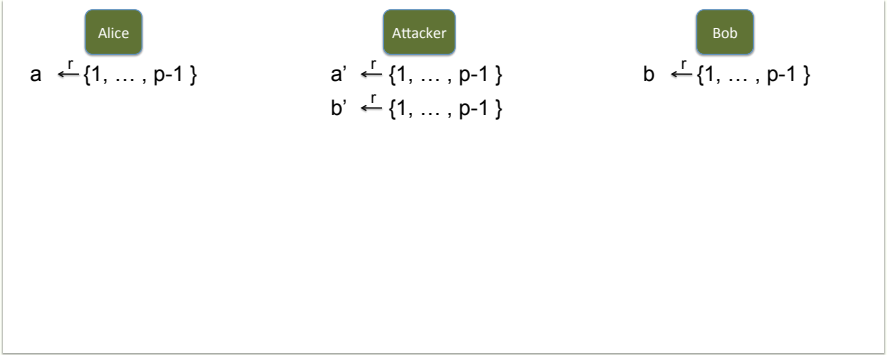


The Diffie-Hellman (DH) protocol

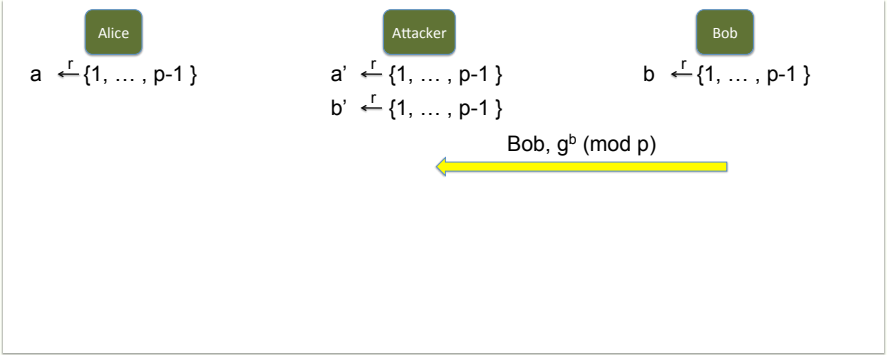
- Assumption: the DHP is hard in $\mathbb{Z}_p^*$
- Fix a very large prime p, and $g \in \{1, \dots, p\text{-}1\}$



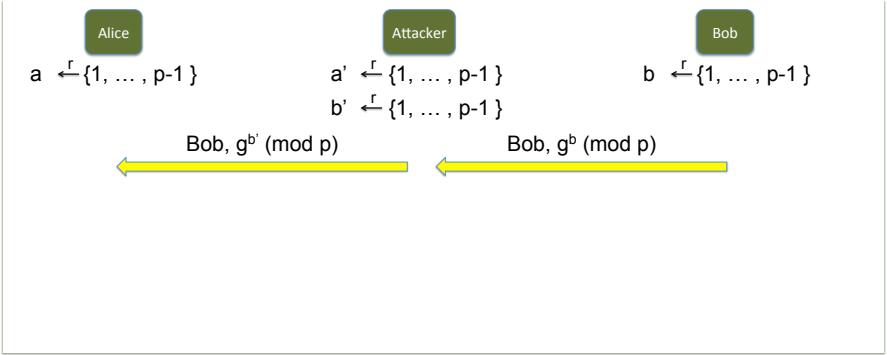
Man in the middle attack on DH



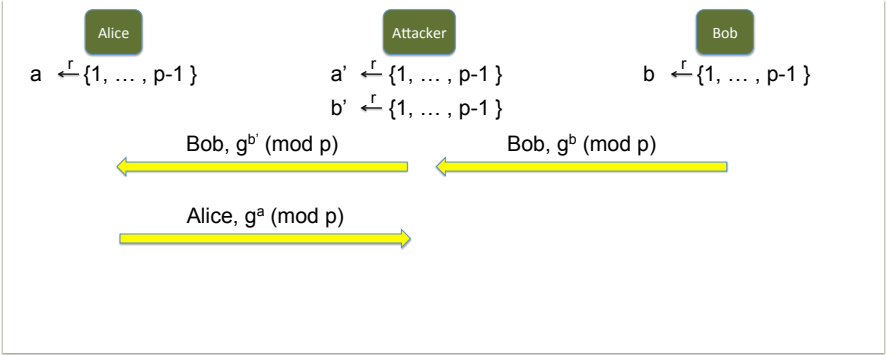
Man in the middle attack on DH



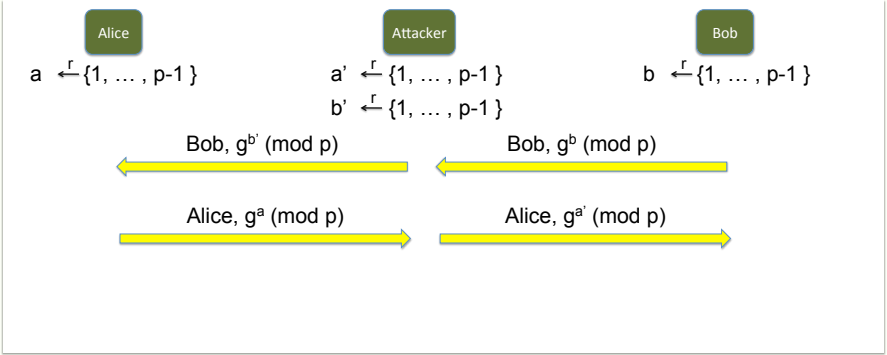
Man in the middle attack on DH



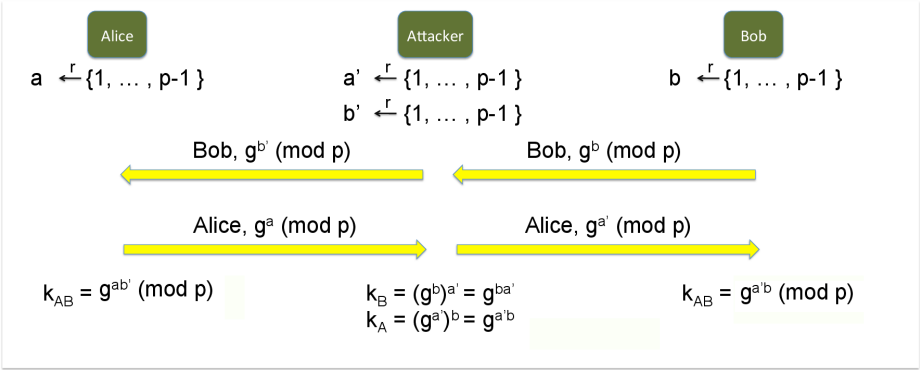
Man in the middle attack on DH



Man in the middle attack on DH



Man in the middle attack on DH



RSA trapdoor permutation

- $G_{RSA}() = (pk, sk)$ where $pk = (N, e)$ and $sk = (N, d)$ and $N = p \cdot q$ with p, q random primes and $e, d \in \mathbb{Z}$ st. $e \cdot d \equiv 1 \pmod{\phi(N)}$
- $\mathcal{M} = \mathcal{C} = \mathbb{Z}_N$
- $RSA(pk, x) = x^e \pmod N$ where $pk = (N, e)$
- $RSA^{-1}(sk, x) = x^d \pmod N$ where $sk = (N, d)$
- Consistency: $\forall (pk, sk) = G_{RSA}(), \forall x, RSA^{-1}(sk, RSA(pk, x)) = x$

Proof: Let $pk = (N, e)$, $sk = (N, d)$. and $x \in \mathbb{Z}_N$. Easy case where x and N are relatively prime

$$\begin{aligned} RSA^{-1}(sk, RSA(pk, x)) &= (x^e)^d \pmod N \\ &= x^{e \cdot d} \pmod N \\ &= x^{1 + k\phi(N)} \pmod N \\ &= x \cdot x^{k\phi(N)} \pmod N \\ &= x \cdot (x^{\phi(N)})^k \pmod N \\ &\overset{\text{Euler}}{=} x \pmod N \end{aligned}$$

How NOT to use RSA

(G_{RSA}, RSA, RSA^{-1}) is called raw RSA (or sometimes Textbook RSA).

Do not use raw RSA directly as an asymmetric cipher!

RSA is deterministic and malleable $\Rightarrow$ not secure against chosen plaintext attacks

ISO standard: ISO/IEC 18033-2

Goal: build a CPA secure asymmetric cipher using (G_{RSA}, RSA, RSA^{-1})

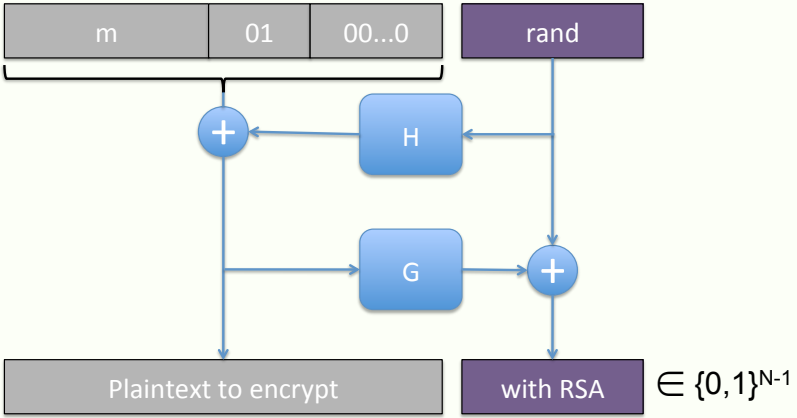
Let (E_s, D_s) be a symmetric encryption scheme over $(\mathcal{M}, \mathcal{C}, \mathcal{K})$
Let $H : \mathbb{Z}_N^* \to \mathcal{K}$

Build $(G_{RSA}, E_{RSA}, D_{RSA})$ as follows

- $G_{RSA}()$ as described above
- $E_{RSA}(pk, m)$:
  - pick random $x \in \mathbb{Z}_N^*$
  - $y \leftarrow RSA(pk, x)$ $(= x^e \mod N)$
  - $k \leftarrow H(x)$
  - return $y || E_s(k, m)$
- $D_{RSA}(sk, y||c) = D_s(H(RSA^{-1}(sk, y)), c)$

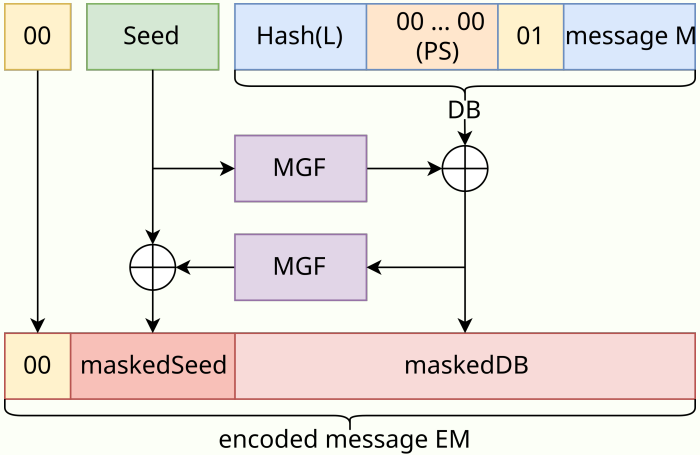
PKCS1 v2.0: RSA-OAEP (old version)

Goal: build a CCA secure asymmetric cipher using (G_{RSA}, RSA, RSA^{-1})



PKCS1 v2.0: RSA-OAEP (wiki version)

Goal: build a CCA secure asymmetric cipher using (G_{RSA}, RSA, RSA^{-1})



ElGamal (EG)

- Fix prime p, and generator $g \in \mathbb{Z}_p^*$
- $\mathcal{M} = \{0, \dots, p\text{-}1\}$ and $\mathcal{C} = \mathcal{M} \times \mathcal{M}$
- $G_{EG}() = (pk, sk)$ where $pk = g^d \pmod p$ and $sk = d$ and $d \xleftarrow{r} \{1, \dots, p\text{-}2\}$
- $E_{EG}(pk, x) = (g^r \pmod p,\ m \cdot (g^d)^r \pmod p)$ where $pk = g^d \pmod p$ and $r \xleftarrow{r} \mathbb{Z}$
- $D_{EG}(sk, x) = e^{\text{-}d} \cdot c \pmod p$ where $x = (e, c)$
- Consistency: $\forall (pk, sk) = G_{EG}(), \forall x, D_{EG}(sk, E_{EG}(pk, x)) = x$

Proof: Let $pk = g^d \pmod p$ and $sk = d$

$$\begin{aligned} D_{EG}(sk, E_{EG}(pk, x)) &= (g^r)^{\text{-}d} \cdot m \cdot (g^d)^r \pmod p \\ &= m \pmod p \end{aligned}$$

Computer Security

INFR10067

Fall 2025

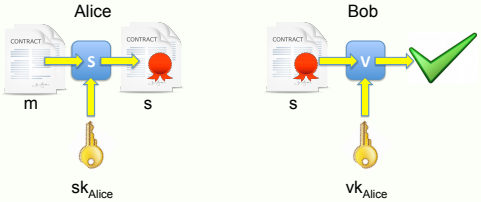
Cryptography

Digital Signatures

Myrto Arapinis and **Markulf Kohlweiss**
School of Informatics
University of Edinburgh

Goal

Data integrity and origin authenticity in the public-key setting



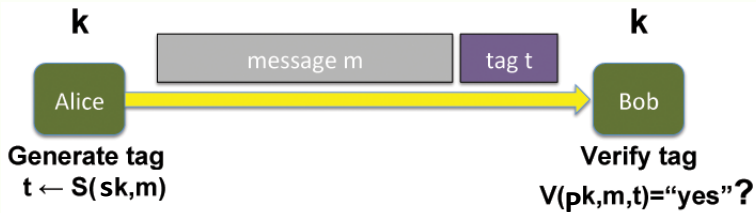
- key generation algorithm: $G : \rightarrow \mathcal{K}_{pk} \times \mathcal{K}_{sk}$
- signing algorithm $S : \mathcal{K}_{sk} \times \mathcal{M} \rightarrow \mathcal{S}$
- verification algorithm $V : \mathcal{K}_{pk} \times \mathcal{M} \times \mathcal{S} \rightarrow \{1, 0\}$
- s.t. $\forall (sk, vk) \in G$, and $\forall m \in \mathcal{M}$, $V(vk, m, S(sk, m)) = 1$

Advantages of digital signatures over MACs

MACs

- are not publicly verifiable (and so not transferable)
  No one else, except Bob, can verify t.
- do not provide non-repudiation
  t is not bound to Alice's identity only. Alice could later claim she didn't compute t herself. It could very well have been Bob since he also knows the key k.

Advantages of digital signatures over MACs



Digital signatures

- are publicly verifiable - anyone can verify a signature
- are tansferable - due to public verifiability
- provide non-repudiation - if Alice signs a document with sk, she cannot deny it later
  Alice is responsible for keeping her secret key sk secret.

Security

A good digital signature schemes should satisfy existential unforgeability.

Existential unforgeability

- Given $(m_1, S(sk, m_1))$, ..., $(m_n, S(sk, m_n))$ (where m_1, ..., m_n chosen by the adversary)
- It should be hard to compute a valid pair $(m, S(sk, m))$ without knowing sk for any $m \notin \{m_1, \dots, m_n\}$

Textbook RSA signatures

- $G_{RSA}() = (pk, sk)$ where $pk = (N, e)$ and $sk = (N, d)$ and $N = p \cdot q$ with p, q random primes and $e, d \in \mathbb{Z}$ st. $e \cdot d \equiv 1 \pmod{\phi(N)}$
- $\mathcal{M} = \mathcal{C} = \mathbb{Z}_N$
- Signing: $S_{RSA}(sk, x) = (x, x^d \pmod N)$ where $pk = (N, e)$
- Verifying: $V_{RSA}(pk, m, x) = \begin{cases} 1 & \text{if } m = x^e \pmod N \\ 0 & \text{otherwise} \end{cases}$
  where $sk = (N, d)$
- st $\forall (pk, sk) = G_{RSA}(), \forall x, V_{RSA}(pk, x, S_{RSA}(sk, x)) = 1$
  Proof: exactly as proof of consistency of RSA encryption/decryption

Problems with “textbook RSA signatures”

Textbook RSA signatures are not secure

The “textbook RSA signature” scheme does not provide existential unforgeability

- Suppose Eve has two valid signatures $\sigma_1 = M_1^d \mod N$ and $\sigma_2 = M_2^d \mod N$ from Bob, on messages M_1 and M_2.
- Then Eve can exploit the homomorphic properties of RSA and produce a new signature

$$\sigma = \sigma_1 \cdot \sigma_2 \mod N = M_1^d \cdot M_2^d \mod N = (M_1 \cdot M_2)^d \mod N$$

which is a valid signature from Bob on message $M_1 \cdot M_2$.

How to use RSA for signatures

Solution

Before computing the RSA function, apply a hash function H.

- Signing: $S_{RSA}(sk, x) = (x, H(x)^d \pmod N)$
- Verifying: $V_{RSA}(pk, m, x) = \begin{cases} 1 & \text{if } H(m) = x^e \pmod N \\ 0 & \text{otherwise} \end{cases}$

Computer Security

INFR10067

Fall 2025

Secure communications

Public Key Infrastructure

Myrto Arapinis and **Markulf Kohlweiss**
School of Informatics
University of Edinburgh

Public keys

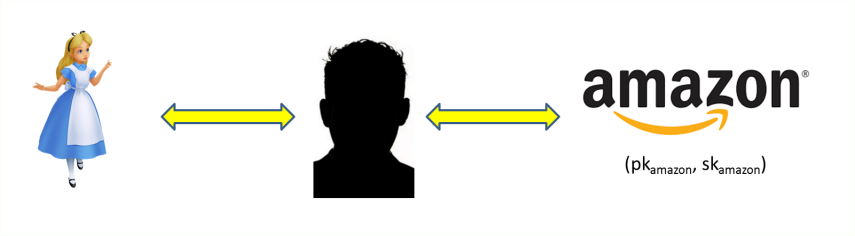


Figure: How does Alice trust that pk_{Amazon} is Amazon's public key?

Public-key encryption schemes are secure only if the authenticity of the public key is assured

Distribution of public keys

1. Public announcements - participants broadcast their public key
   ☹ does not defend against fake certificates
2. Publicly available directories - participants publish their public key on public directories
   ☹ does not defend against fake certificates
3. Public-key authority - participants contact the authority for each public key it needs
   ☹ bottleneck in the system
4. public-key certificates - CAs issue certificates to participants on their public key
   ☺ as reliable as public-key authority but avoiding the bottleneck

Critical step: Certificate validation

- **The crux of TLS security**: Server authenticates itself to client
- **What is a certificate?**
  - A public key + signature of that key
    - Encoded in an insanely asinine byte format
  - Associated information (domain name, organization, etc.)
  - Essentially says: “I am google.com, and my public key is [key]”
- **Certificate validation process**:
  1. Browser receives certificate from server
  2. Verifies the certificate’s signature
  3. If signature is valid →certificate and public key are trusted
  4. Browser proceeds with connection

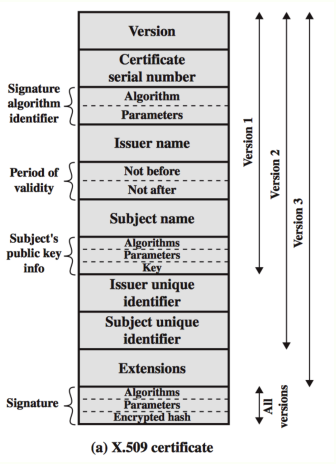


Figure: Image from Cryptography and Network Security - Principles and Practice - William Stallings

Certificate Authorities: The trust foundation

- **Problem**: How does the browser know which public key to use for verification?
- **Solution**: Certificate Authorities (CAs)
  - Public keys hardcoded in your browser/OS.
  - Trusted organizations that issue certificates.
  - Act as trusted third parties.
- **CA's role**:
  - Verify that public keys belong to claimed entities.
  - Sign certificates with their private keys.
  - Protect their private keys from compromise.
- **Without CAs**: No way to distinguish legitimate servers from MITM attackers!

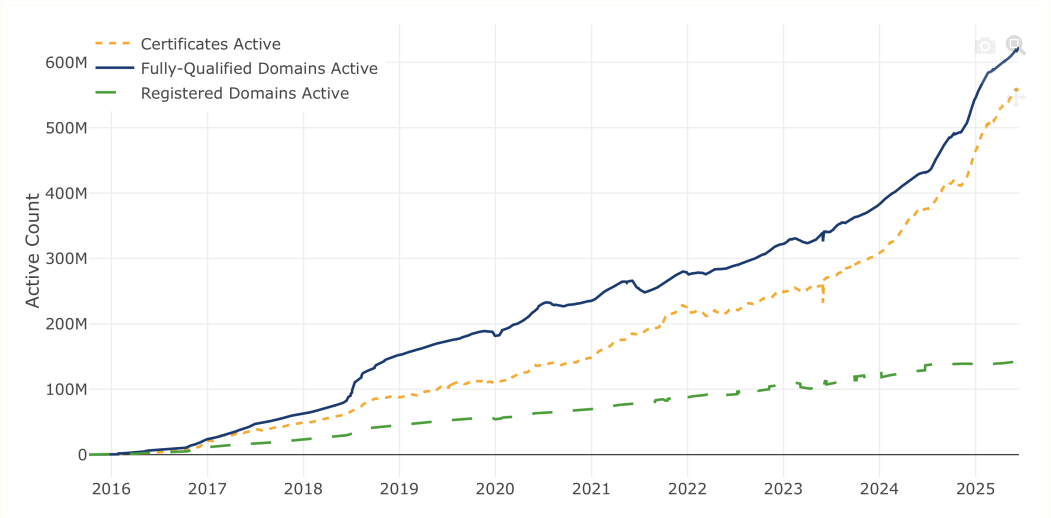
Traditional CA validation nightmare

- **Getting a certificate before 2015**:
  - Contact a Certificate Authority (Verisign, Thawte, etc.).
  - Pay $50-300+ per certificate per year.
    - Completely arbitrary extortionate amounts.
  - Manual validation process takes days/weeks.
- **Domain Validation (DV) process**:
  1. Submit CSR (Certificate Signing Request).
  2. CA sends email to `admin@domain.com`.
  3. Click verification link in email.
  4. Wait for manual review and approval.
  5. Download and install certificate.
- **Extended Validation (EV) certificates**:
  - Even more expensive ($150-1000+/year).
  - Weeks of back-and-forth communication.

Let’s Encrypt: revolution in TLS certificates (2015)

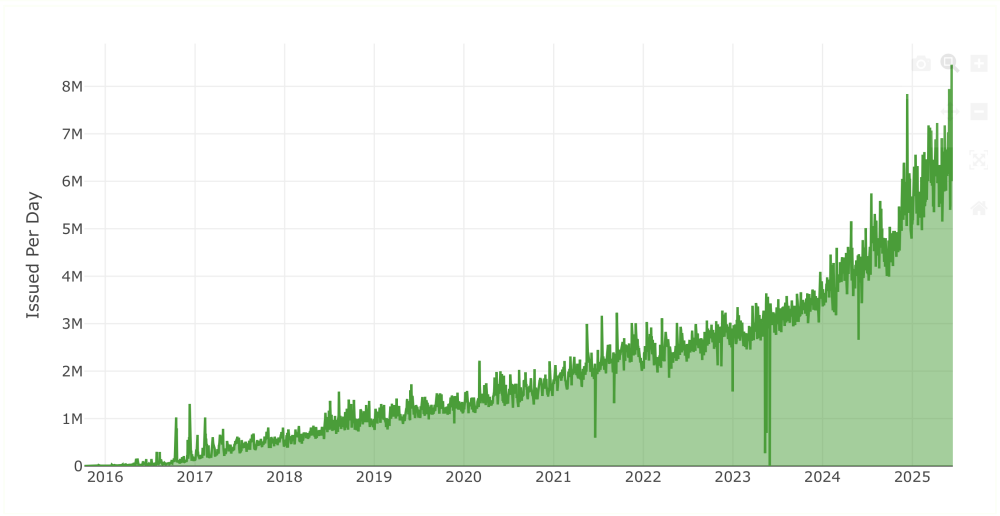
- **Game-changing**:
  - **Free** certificates for everyone.
  - **Automated** issuance and renewal.
  - **90-day** certificate lifetime (encourages automation).
  - Open source, non-profit initiative.
- **ACME protocol** (Automated Certificate Management Environment):
  - Domain validation via HTTP or DNS challenges.
  - Prove control of domain programmatically.
  - Certificate issued in seconds, not days.
  - Automatic renewal before expiration.
- **Impact on the web**:
  - HTTPS adoption jumped from 40% to 90%+ of web traffic.
  - Eliminated cost barrier for small websites.
  - Made “HTTPS everywhere” a reality.
- **Philosophy**: Encryption should be the default, not a luxury.

Let’s Encrypt: active certificates



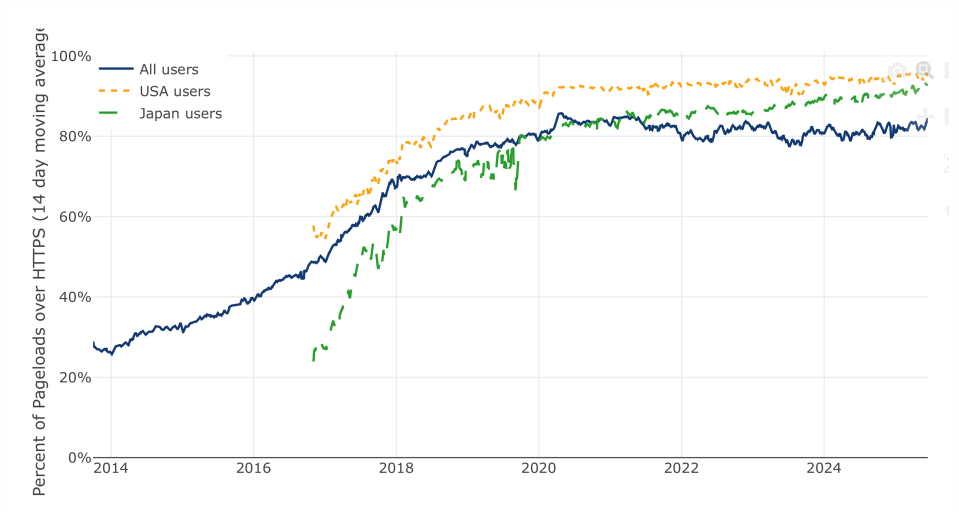
Source: Let’s Encrypt

Let’s Encrypt: certificates issued



Source: Let’s Encrypt

Percentage of HTTPS traffic in Firefox



Source: Let's Encrypt

Certificate chains in practice

- **Real-world example**: Connecting to `www.google.com`
- **Certificate chain structure**:
  - **Certificate 0**: `www.google.com`'s certificate.
  - **Certificate 1**: Intermediate CA (Google Trust Services).
  - **Certificate 2**: Root CA (GlobalSign).
- **Verification process**:
  1. Check Google's signature on Certificate 0.
  2. Check GlobalSign's signature on Certificate 1.
  3. Trust GlobalSign's root certificate (pre-installed).
- **Chain of trust**: Each certificate vouches for the next.

Exploring certificates with OpenSSL

- **Connect and view certificate**:
  - `openssl s_client -connect www.google.com:443`
  - Shows certificate chain and raw certificate data
- **Parse certificate details**:
  - `openssl x509 -text -noout`
  - Reveals subject, issuer, validity dates, algorithms
- **Certificate markers**:
  - `s:` = subject (who the certificate is for)
  - `i:` = issuer (who signed the certificate)
- Try this at home! Great way to understand the certificate ecosystem.

```
openssl s_client -connect www.google.com:443 -servername
www.google.com | openssl x509 -text -noout
```

Public key certificates

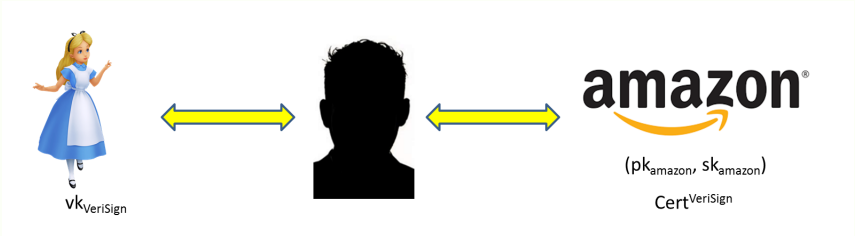
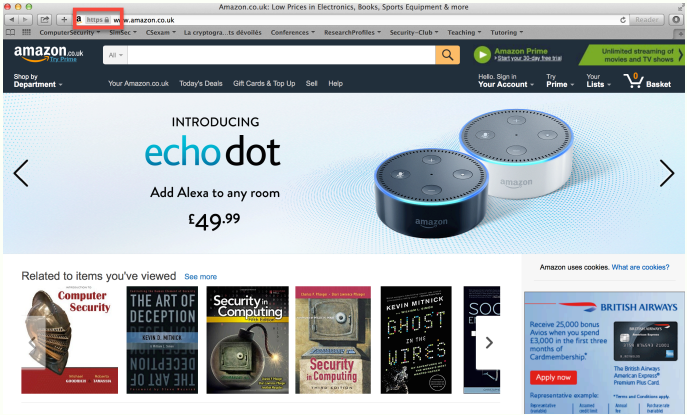


Figure: Alice can now verify Amazon's certificate

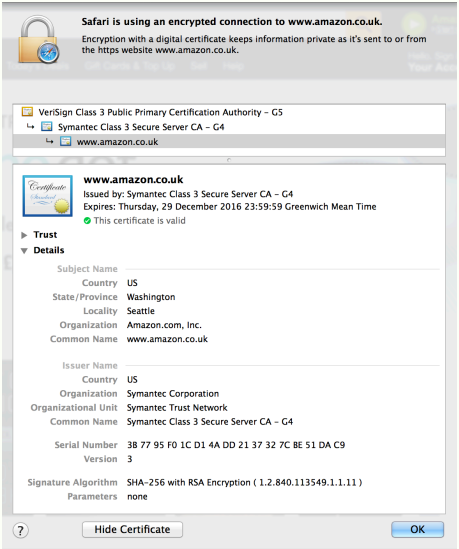
Using public key certificates to secure the Internet



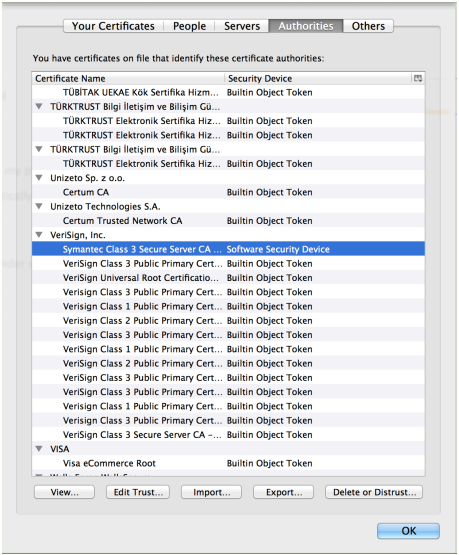
A very important implicit assumption

The browser is trusted to be “secure”

Amazon's certificate



Browser root certificates



Revocation

- A certificate needs to be revoked if the corresponding private key has been compromised.
- Certificate Revocation Lists (CRLs) are the solution adopted in X.509.
- Online Certificate Status Protocol (OCSP) stapling is the modern solution to this problem.