Cryptography: Introduction

Myrto Arapinis
School of Informatics
University of Edinburgh

January 21, 2019

What is cryptography?

Concise Oxford Dictionary definition

Cryptography is the art of writing and solving codes

But nowadays cryptography encompasses many more things than just secret communications: message authentication, digital signatures, protocols for exchanging secret keys, authentication protocols, anonymous communications, electronic auctions and elections, digital cash, *etc.*

> *"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

What is cryptography?

It is:

- A very powerful tool for confidentiality, integrity, authentication, anonymity, *etc.*
- Everywhere, as the core of many security mechanisms.

It is not:

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

Cryptography is everywhere!

Cryptographic methods are used:

- to securely access a website;
- when talking over a mobile phone;
- to enforce access control in a multi-user operating system;
- to prevent thieves extracting trade secrets from stolen laptops;
- to prevent software copying;
- *etc*

Cryptography (and security more broadly) is becoming a more and more central topic within computer science

In this course

We present only the rudiments of the topic:

- what cryptography can achieve;
- that it can go wrong;
- what is good practice when using it.

We will discuss constructions for:

- Symmetric encryption;
- Asymmetric (public) encryption;
- Hash functions and MACs;
- Digital signatures.

Historical ciphers

Rail fence cipher

- shared secret key $k \in \mathbb{N}$
- Encryption: plaintext written in columns of size k. The ciphertext is the concatenation of the resulting rows.

k=6

m = THIS COURSE AIMS TO INTRODUCE YOU TO THE PRINCIPLES AND TECHNIQUES OF SECURING COMPUTERS

T	O	A	O	O	Y		R	L	D	N		C		T
H	U	I		D	O	T	I	E		I	O	U	C	E
I	R	M	I	U	U	H	N	S	T	Q	F	R	O	R
S	S	S	N	C		E	C		E	U		I	M	S
	E		T	E	T		I	A	C	E	S	N	P	
C		T	R		O	P	P	N	H	S	E	G	U	

c = TOAOOY RLDN C THUI DOTIE IOUCEIRMIUUHNSTQFRORSSSNC EC EU IMS E TET IACESNPC TR OPPNHSEGUQ

- Decryption: ciphertext written in rows of size $\frac{|c|}{k}$

But small key space size: $k < |c| \Rightarrow$ brute force attack!!

Substitution cipher

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

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

- Encryption: apply ϖ to each character of the plaintext.

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

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

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

Substitution cipher: example

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

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.

Breaking the substitution cipher

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

    $e > t > a > o$
  - Use frequency of digrams in English text

    $th > he > in > er$
  - Use frequency of trigrams in English text

    $the > and > ing$
  - Use expected words

Breaking the substitution cipher: example

$\varpi =$

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

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

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

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

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 digrams in c: of $>$ zi $>$ ...
t$\mapsto$z suggests h$\mapsto$i

Breaking the substitution cipher: example

$\varpi = \quad t \mapsto z\ e \mapsto t\ h \mapsto i$

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$\mapsto$of

Breaking the substitution cipher: example

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

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

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

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$\mapsto$k

Breaking the substitution cipher: example

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

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

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

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$\mapsto$l

Breaking the substitution cipher: example

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

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

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

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

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

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!!

Vigenere cipher

- shared secret key: a word w over the English alphabet
- Encryption: break the plaintext $m = m_1 \dots m_n$ in $\frac{|m|}{|w|}$ blocks, and encrypt each block as follows

$$\begin{array}{cccc} & m_{i+1} & \dots & m_{i+|w|} \\ + & w_1 & \dots & w_{|w|} \\ \hline & \underbrace{m_{i+1} + w_1 \pmod{26}}_{c_{i+1}} & \dots & \underbrace{m_{i+|w|} + w_{|w|} \pmod{26}}_{c_{i+|w|}} \end{array}$$

  Concatenate the resulting blocks to obtain the ciphertext
- Decryption: break the ciphertext $c = c_1 \dots c_n$ in $\frac{|m|}{|w|}$ blocks, and decrypt each block as follows

$$\begin{array}{cccc} & c_{i+1} & \dots & c_{i+|w|} \\ - & w_1 & \dots & w_{|w|} \\ \hline & \underbrace{c_{i+1} - w_1 \pmod{26}}_{m_{i+1}} & \dots & \underbrace{c_{i+|w|} - w_{|w|} \pmod{26}}_{m_{i+|w|}} \end{array}$$

  Concatenate the resulting blocks to retrieve the message

Vigenere cipher

- shared secret key w a word
- Encryption: break the plaintext $m = m_1 \dots m_n$ in $\frac{|m|}{|w|}$ blocks, and encrypt each block as follows

$$\begin{array}{cccc} & m_{i+1} & \dots & m_{i+|w|} \\ + & w_1 & \dots & w_{|w|} \\ \hline & \underbrace{m_{i+1} + w_1 \pmod{26}}_{c_{i+1}} & \dots & \underbrace{m_{i+|w|} + w_{|w|}}_{c_{i+|w|} \pmod{26}} \end{array}$$

  Concatenate the resulting blocks to obtain the ciphertext

- Decryption: break the ciphertext $c = c_1 \dots c_n$ in $\frac{|m|}{|w|}$ blocks, and encrypt each block as follows

$$\begin{array}{cccc} & c_{i+1} & \dots & c_{i+|w|} \\ - & w_1 & \dots & w_{|w|} \\ \hline & \underbrace{c_{i+1} + w_1 \pmod{26}}_{m_{i+1}} & \dots & \underbrace{c_{i+|w|} + w_{|w|}}_{m_{i+|w|} \pmod{26}} \end{array}$$

  Concatenate the resulting blocks to retrieve the message

Vigenere cipher: example

w = MACBETH

m = WHEN SHALL WE THREE MEET AGAIN IN THUNDERLIGHTNING OR IN RAIN

	M	A	C	B	E	T	H	M	A	C	B	E	T	H	. . .	M	A
+	W	H	E	N	S	H	A	L	L	W	E	T	H	R	. . .	I	N
	J	H	G	O	W	A	H	X	L	Y	F	X	A	Y	. . .	U	N

c = JHGO WAHXL YF XAYQE OFIM HSAKO MG ATUPEIKSUGJURBUS OT JR KHUN

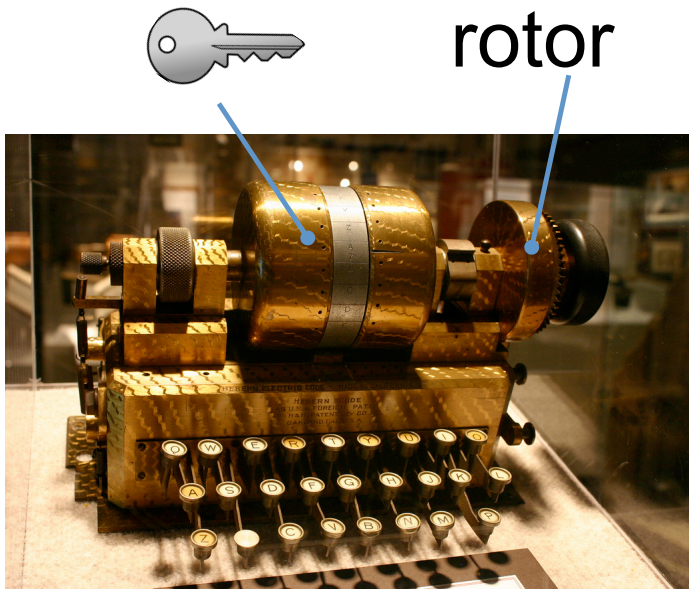
Breaking the Vigenere cipher

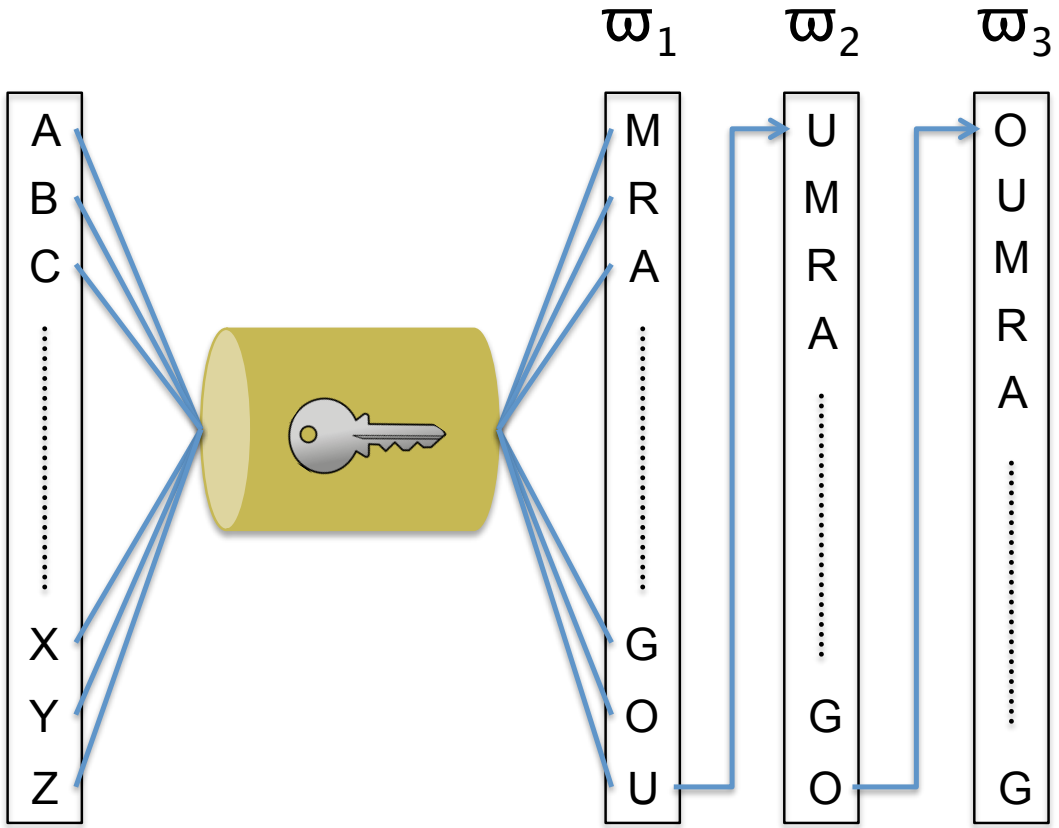
- Suppose we know the length of the key w. Break the ciphertext in $\frac{|c|}{|w|}$ blocks:

$$c_1 \dots c_{|w|} \;||\; c_{|w|+1} \dots c_{2|w|} \;||\; \dots \;||\; c_{|c|-|w|+1} \dots c_{|c|}$$

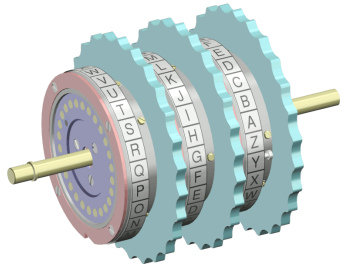
  for each position in $\{1, \dots, |w|\}$, consider the characters $c_{j|w|+i}$ for all $j \in \frac{|c|}{|w|}$. All these characters have been encrypted using the same key character w_i. Perform letter frequency analysis on this set of characters.
- If the size of w is not known apply Kasiski's method to narrow the possibilities:
  - identify all the sequences of letters of length greater than 4 that occur more than once
  - for each such sequence compute the distance between two of its occurrences
  - compute the corresponding possible key-length

Rotor machines: the Herbern machine

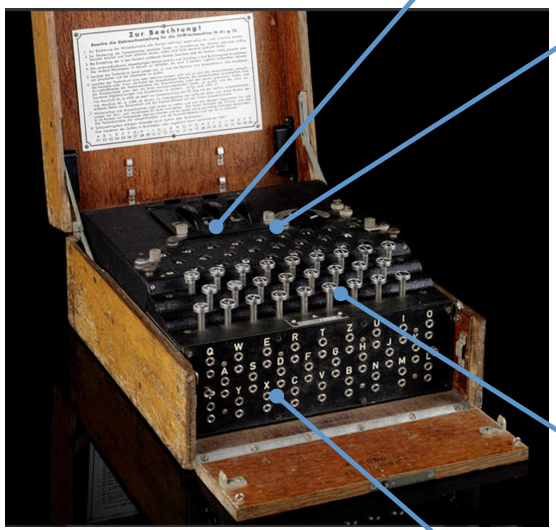


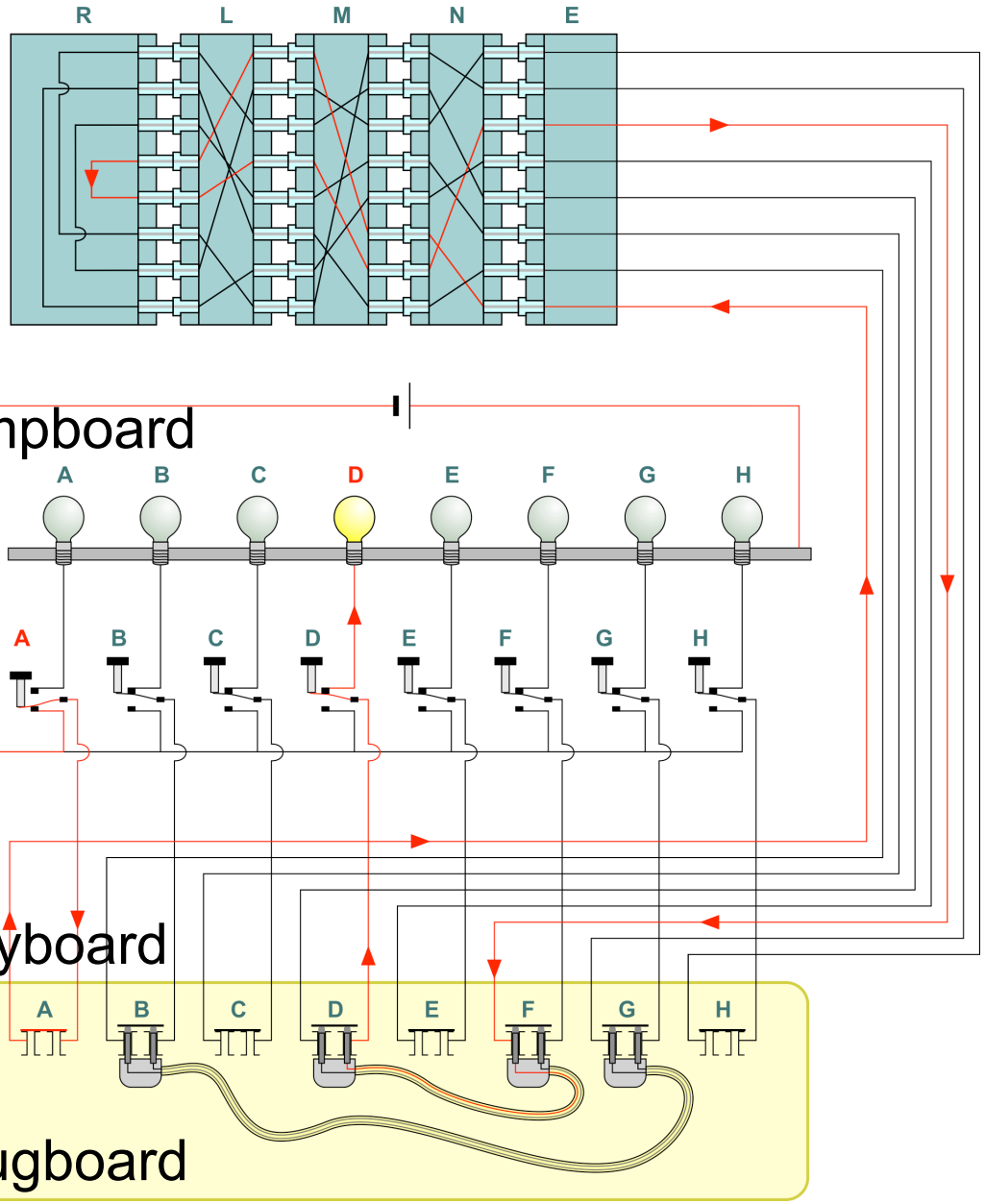


Rotor machines: the enigma machine



Rotors





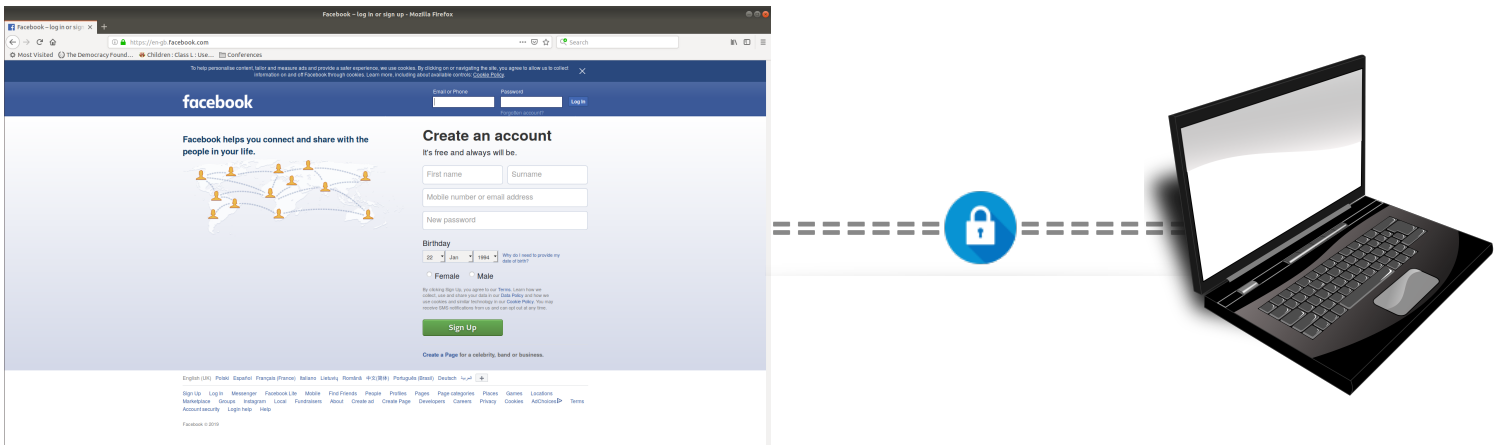
Cryptography: Symmetric encryption

Myrto Arapinis
School of Informatics
University of Edinburgh

January 23, 2019

Goal: confidentiality

- Secure communications



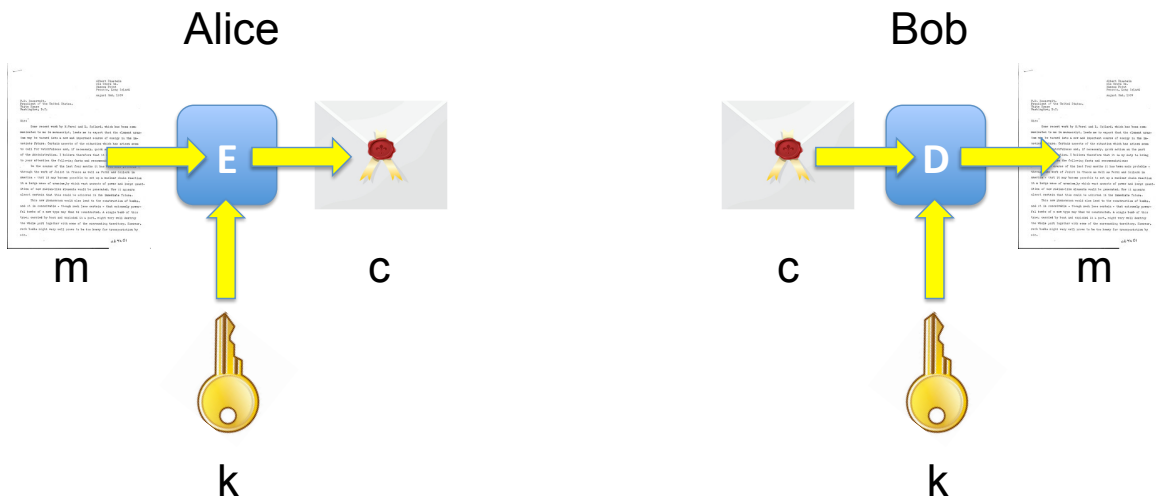
- File protection

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 archeture 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 scutinised by many users, white hat hackers, academics, *etc.*
$\rightsquigarrow$ early discovery and corrections of flaws/vulnerabilities

Adversarie'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 may have access to :

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

Brute-force attack

- 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 $\mathcal{K}$

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

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

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

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 $c \in \mathcal{C}$

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

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

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

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.
- 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 $m' = to\ eve : m_0$
    $c' := c \oplus$ "$to\ bob : 00 \ldots 00$" $\oplus$ "$to\ eve : 00 \ldots 00$"

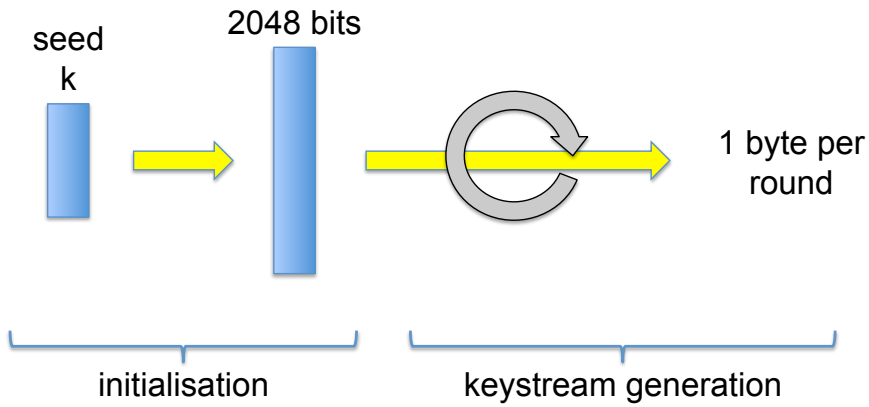
Stream ciphers

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)
    $G: \{0,1\}^s \rightarrow \{0,1\}^n$ 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:



- Main data structure: array S of 256 bytes.
- Used in HTTPS and WEP
- Weaknesses of RC4:
  - first bytes are biased
    $\longrightarrow$ drop the first to 256 generated bytes
  - subject to related keys attacks
    $\longrightarrow$ choose randomly generated keys as seeds

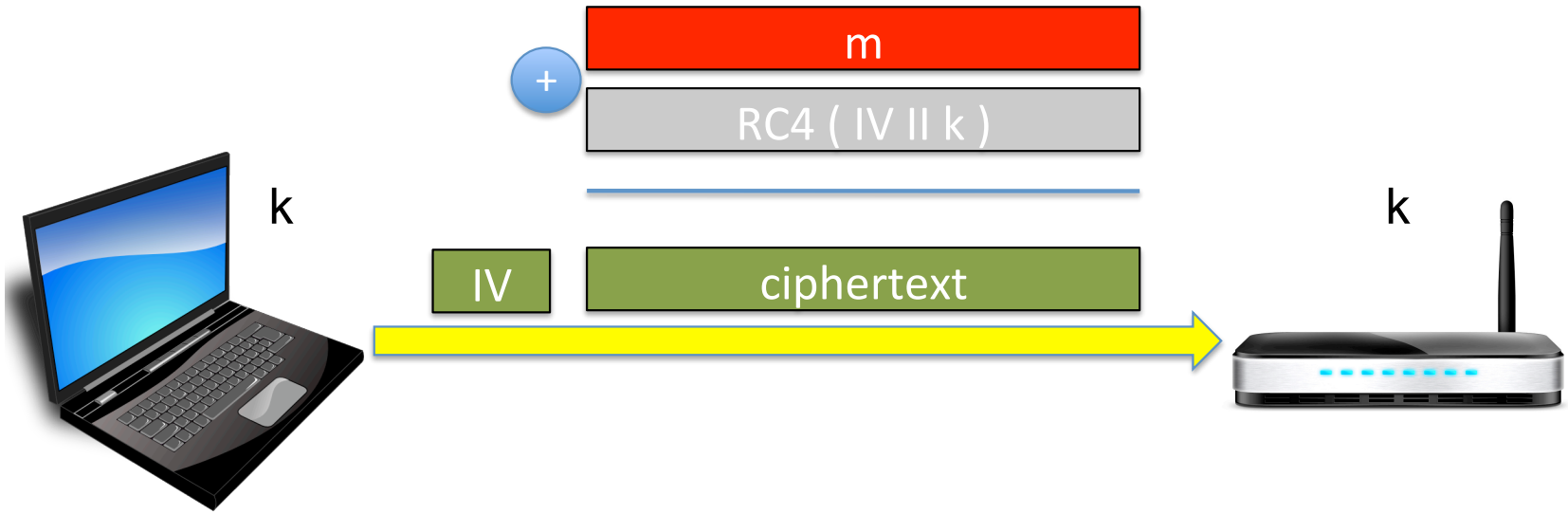
RC4: initialisation

```
for i := 0 to 255 do
    S[i] := i
end
j := 0
for i := 0 to 255 do
    j := (j + S[i] + K[i(mod |K|)])(mod 256)
    swap(S[i], S[j])
end
i := 0
j := 0
```

RC4: key stream generation

```
while generatingOutput
    i := i + 1(mod 256)
    j := j + S[i](mod 256)
    swap(S[i], S[j])
    output(S[S[i] + S[j](mod 256)])
end
```

WEP uses RC4



Initialisation Vector (IV): 24-bits long string

Weaknesses of WEP

- 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

Weaknesses of TLS

MUST READ **THESE TEN CITIES ARE HOME TO THE BIGGEST BOTNETS**

RC4 NOMORE crypto exploit used to decrypt user cookies in mere hours

Websites using RC4 encryption need to change their protocols as exploits using design flaws are now far easier to perform.

By Charlie Osborne for Zero Day | July 20, 2015 -- 10:21 GMT (11:21 BST) | Topic: Security

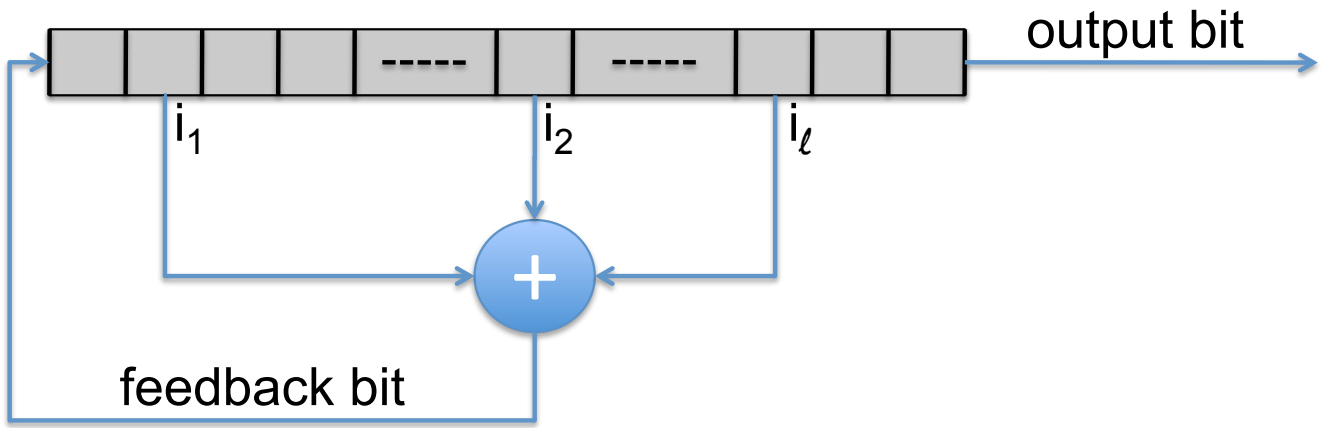
Linear Feedback Shift Registers (LFSRs)

- $\mathcal{K} = \{0,1\}^s$
- Main data structure: register R of s bits
- Initialisation: $R := k$
- Keystream generation: 1-bit output per round

  taps: $i_1, i_2, \ldots i_\ell$

  feedback bit: $R[i_1] \oplus R[i_2] \oplus \cdots \oplus R[i_\ell]$

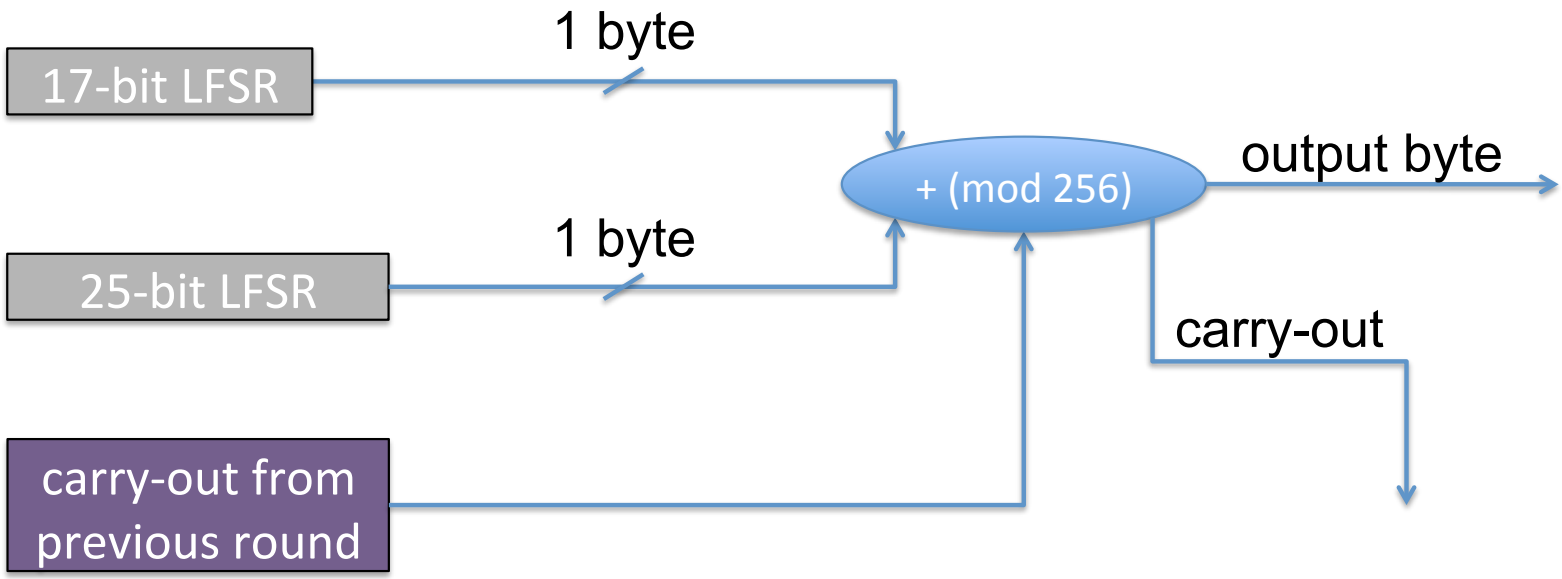
  output bit: $R[s]$



- Broken LFSR-based stream ciphers:
  - DVD encryption: CSS (2 LFSRs)
  - GSM encryption: A5 (3 LFSRs)
  - Bluetooth encryption: E0 (4 LFSRs)

Content Scrambling System (CSS) uses LFSRs

- $\mathcal{K} = \{0,1\}^{40}$
- Data structures: 17-bits LFSR (R_{17}) and 25-bits LFSR (R_{25})
- Initialisation: $R_{17} := 1||K[0-15]$
  $R_{25} := 1||K[16-39]$
- Keystream generation: 1-byte output per round



Weaknesses in CSS

Can be broken in time 2^{17}. The idea of the attack is as follows:

- Because of structure of MPEG-2, first 20 bytes of plaintext are known
- Hence also first 20 bytes of keystream are known
- Given output of 17 bit LFSR, can deduce output of 25 bit LFSR by subtraction
- Hence try all 2^{17} possibilities for 17 bit LFSR and if generated 25 bit LFSR produces observed keystream, cipher is cracked

Android BitCoin attack

ars TECHNICA BIZ & IT TECH SCIENCE POLICY CARS GAMING & CULTURE FORUMS SIGN IN

RANDOM THEFT —

All Android-created Bitcoin wallets vulnerable to theft

Android Java SecureRandom function flaw undermines security of Android wallets.

LEE HUTCHINSON - 8/12/2013, 3:15 PM

Xurxo Martínez

Bitcoin.org released a security advisory over the weekend warning the Bitcoin community that any Bitcoin wallet generated on any Android device is insecure and open to theft. The insecurity appears to stem from a flaw in the Android Java SecureRandom class, which under certain circumstances can

Modern stream ciphers

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

$\longrightarrow$ HC-128, Rabbit, Salsa20/12, SOSEMANUK,
Grain v1, MICKEY 2.0, Trivium

Conjecture

These eStream stream ciphers are "secure"

Concluding remarks

- 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 dues to poor implementations of cryptography

Cryptography: Block ciphers

Myrto Arapinis
School of Informatics
University of Edinburgh

January 25, 2019

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

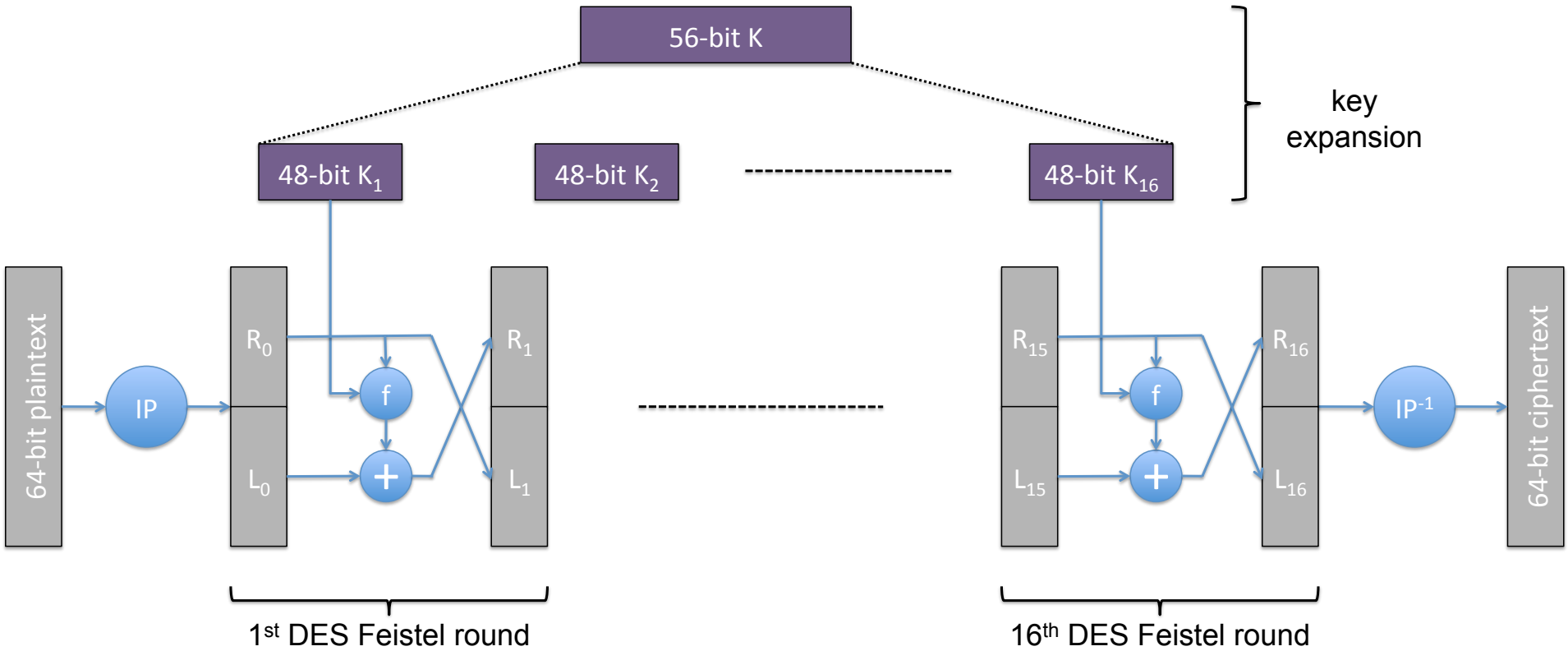
AES: $\ell = 128,\ k = 128, 192, 256$

Data Encryption Standard (DES)

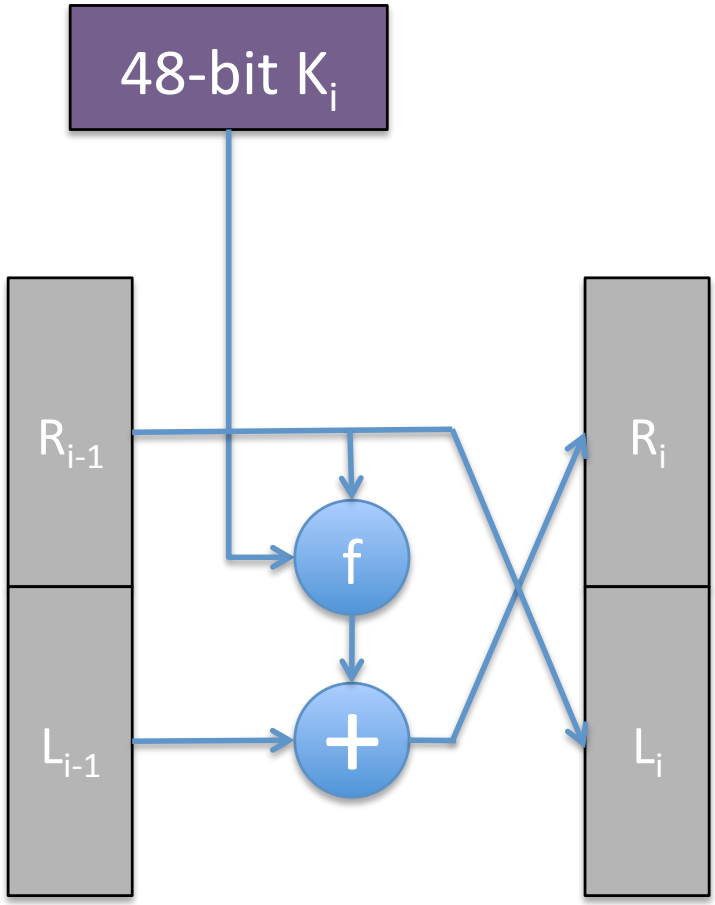
- Early 1970s: Horst Feistel designs Lucifer at IBM
  $k = 128$ bits, $\ell = 128$ bits
- 1973: NBS calls for block cipher proposals.
  $\longrightarrow$ IBM submits a variant of Lucifer.
- 1976: NBS adopts DES as a federal standard
  $k = 56$ bits, $\ell = 64$ bits
- 1997: DES broken by exhaustive search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

Widely deployed in banking (ATM machines) and commerce

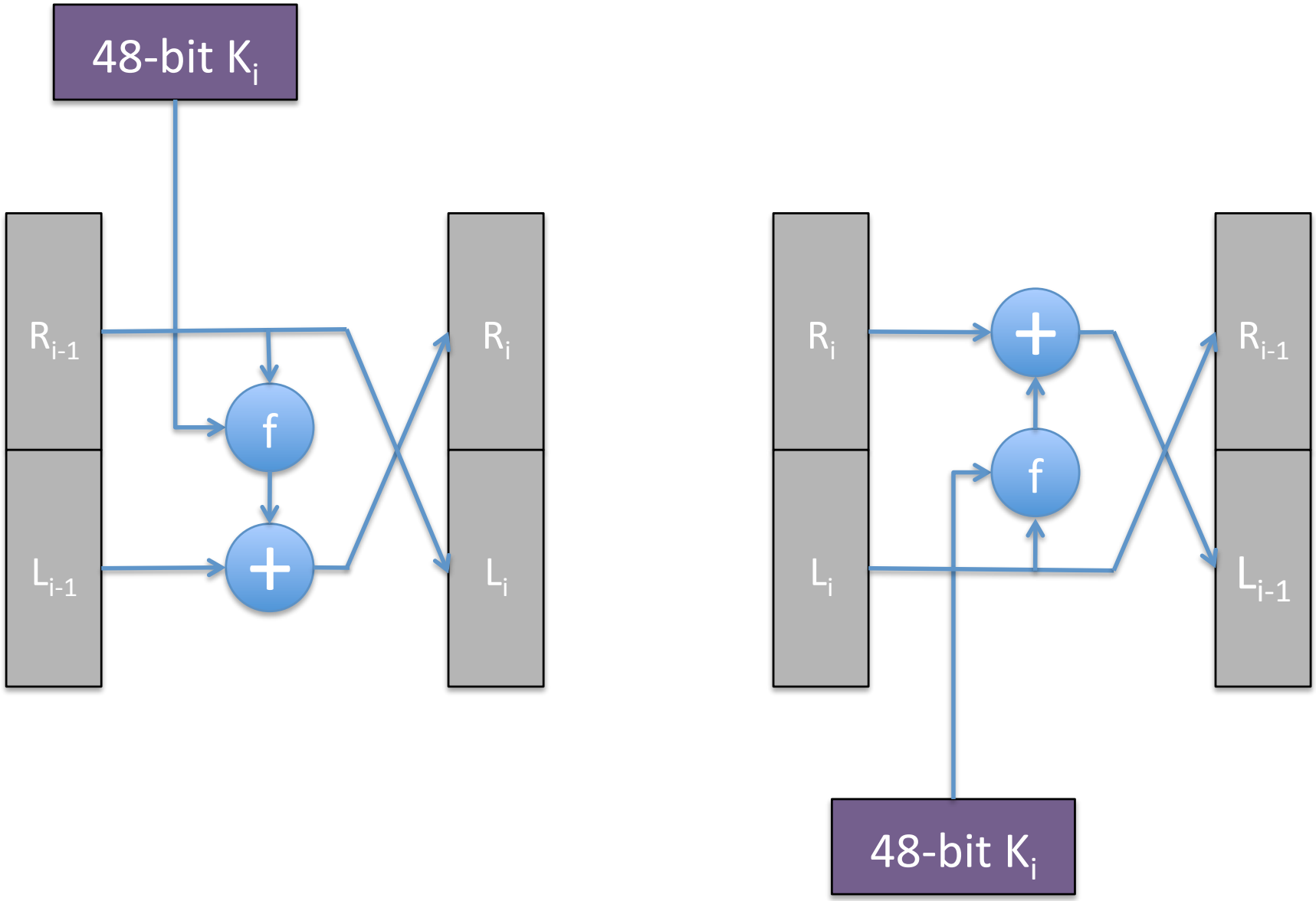
DES: encryption circuit



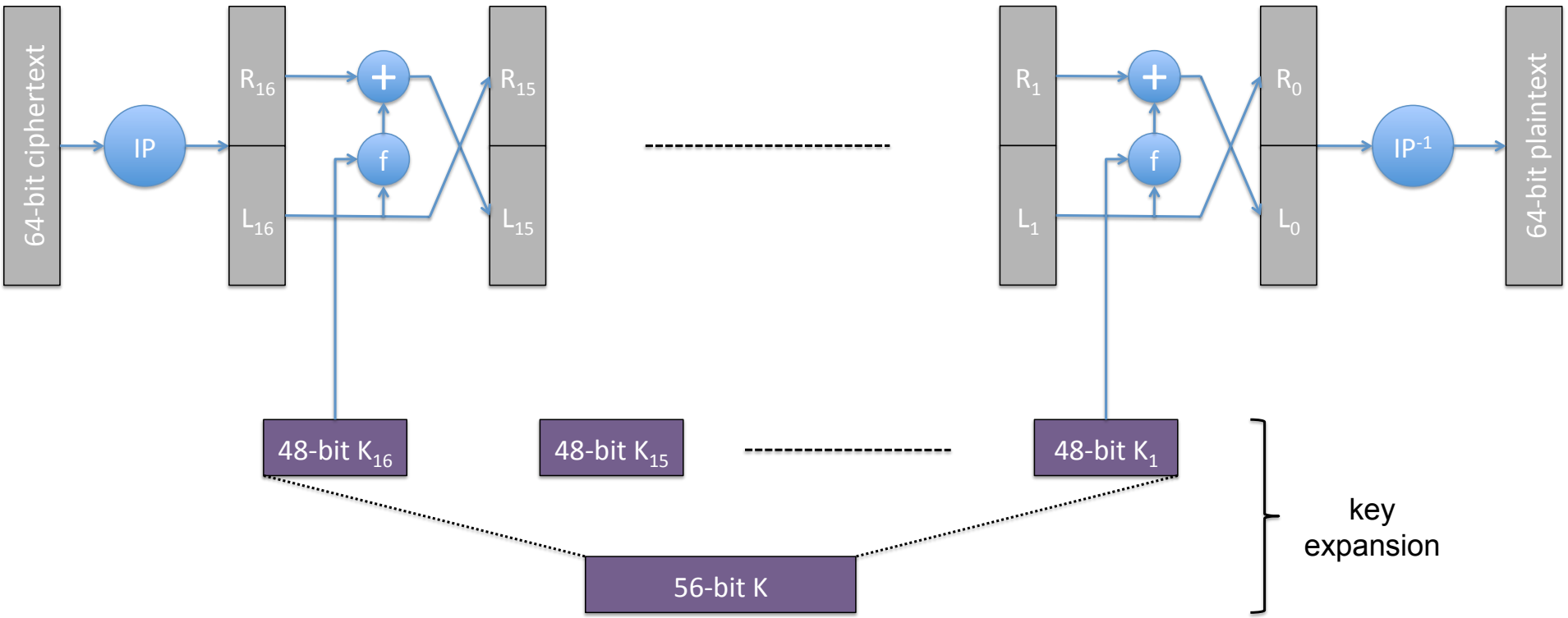
Each DES Feistel round is invertible



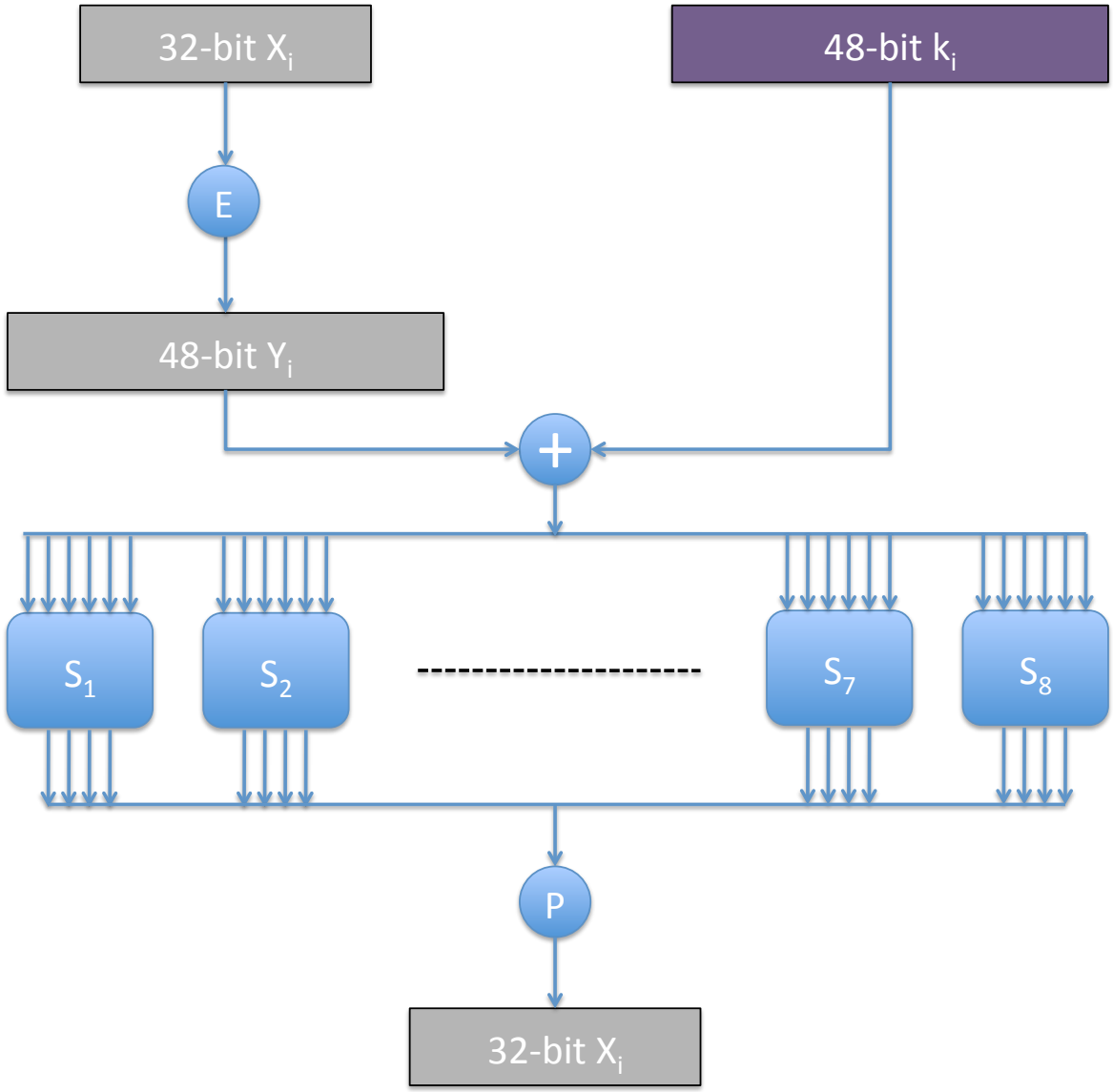
Each DES Feistel round is invertible



DES: decryption circuit



DES: the function f



DES: S_5-box

S_5		Middle 4 bits of input															
		0000	**0001**	**0010**	**0011**	**0100**	**0101**	**0110**	**0111**	**1000**	**1001**	**1010**	**1011**	**1100**	**1101**	**1110**	**1111**
Outer bits	**00**	0010	1100	0100	0001	0111	1010	1011	0110	1000	0101	0011	1111	1101	0000	1110	1001
	01	1110	1011	0010	1100	0100	0111	1101	0001	0101	0000	1111	1010	0011	1001	1000	0110
	10	0100	0010	0001	1011	1010	1101	0111	1000	1111	1001	1100	0101	0110	0011	0000	1110
	11	1011	1000	1100	0111	0001	1110	0010	1101	0110	1111	0000	1001	1010	0100	0101	0011

$$S_5 : \{0,1\}^6 \rightarrow \{0,1\}^4$$

(source: Wikipedia)

$\longrightarrow$ Note that S_5 is not reversible as it maps 6 bits to 4 bits.

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!!

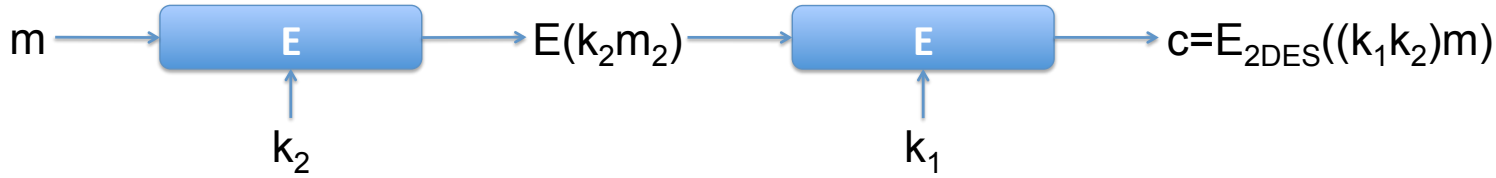
Triple DES (3DES)

- 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)?

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



$\Rightarrow$ For m and c such that $E_{2DES}((k_1, k_2), m) = c$ we have that $E_{DES}(k_2, m) = D_{DES}(k_1, c)$

- 2DES admits a meet-in-the-middle attack that reduces the time for key recovery from 2^{112} for an exhaustive search to 2^{56}. Given $M = (m_1, \ldots, m_{10})$ and $C = (E_{2DES}((k_1, k_2), m_1), \ldots, E_{2DES}((k_1, k_2), m_{10}))$

  - For all possible k_2, $compute E_{DES}(k_2, M)$
  - Sort table according to the resulting $E_{DES}(k_2, M)$

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

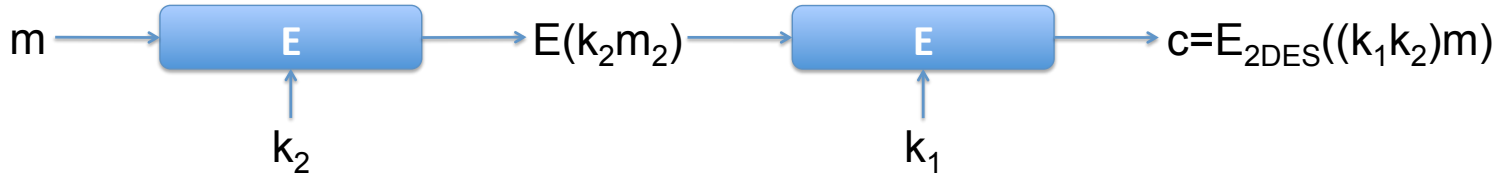
  - For each possible k_1, $compute D_{DES}(k_1, C)$
  - Look up in the table if $D_{DES}(k_1, C) = E_{DES}(k_2, M)$

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

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

What about double DES (2DES)?

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



$\Rightarrow$ For m and c such that $E_{2DES}((k_1, k_2), m) = c$ we have that $E_{DES}(k_2, m) = D_{DES}(k_1, c)$

- 2DES admits a meet-in-the-middle attack that reduces the time for key recovery from 2^{112} for an exhaustive search to 2^{56}. Given $M = (m_1, \ldots, m_{10})$ and $C = (E_{2DES}((k_1, k_2), m_1), \ldots, E_{2DES}((k_1, k_2), m_{10}))$

  - For all possible k_2, $compute E_{DES}(k_2, M)$
  - Sort table according to the resulting $E_{DES}(k_2, M)$

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

  - For each possible k_1, $compute D_{DES}(k_1, C)$
  - Look up in the table if $D_{DES}(k_1, C) = E_{DES}(k_2, M)$

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

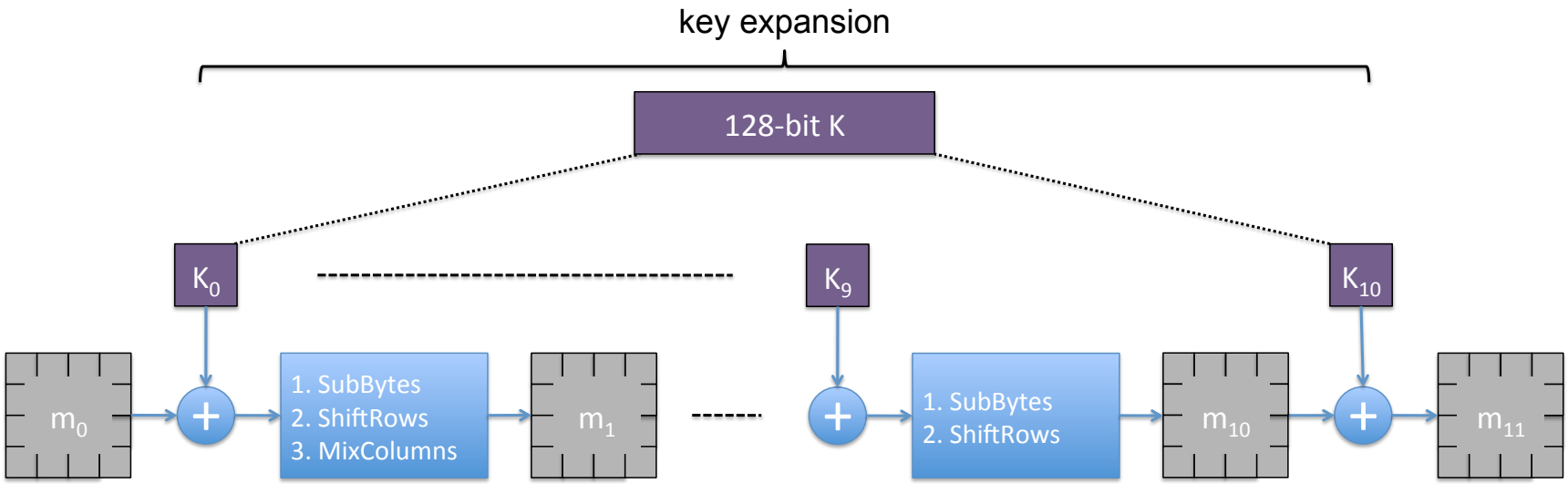
  $\Rightarrow$ time $< 2^{63}$

- Similar attack on 3DES in time 2^{118}

The Advenced 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 is Substitution-Permutation network (not a Feistel network)

AES: encryption circuit



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

$\longrightarrow$ As AES is not a Feistel network, each step needs to be reversible!

AES: SubBytes

	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	xa	xb	xc	xd	xe	xf
0x	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
1x	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
2x	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
3x	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
4x	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
5x	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
6x	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
7x	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
8x	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
9x	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
ax	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
bx	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
cx	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
dx	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
ex	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
fx	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

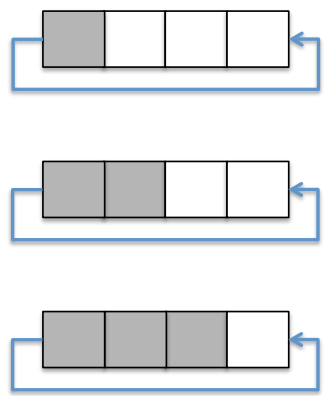
- $\forall j, k.\ m'_i[j,k] = S[m_i[j,k]]$
- rows: most significant 4 bits
- columns: least significant 4 bits

$\longrightarrow$ Note that SubBytes is reversible

AES: ShiftRows

m'_i

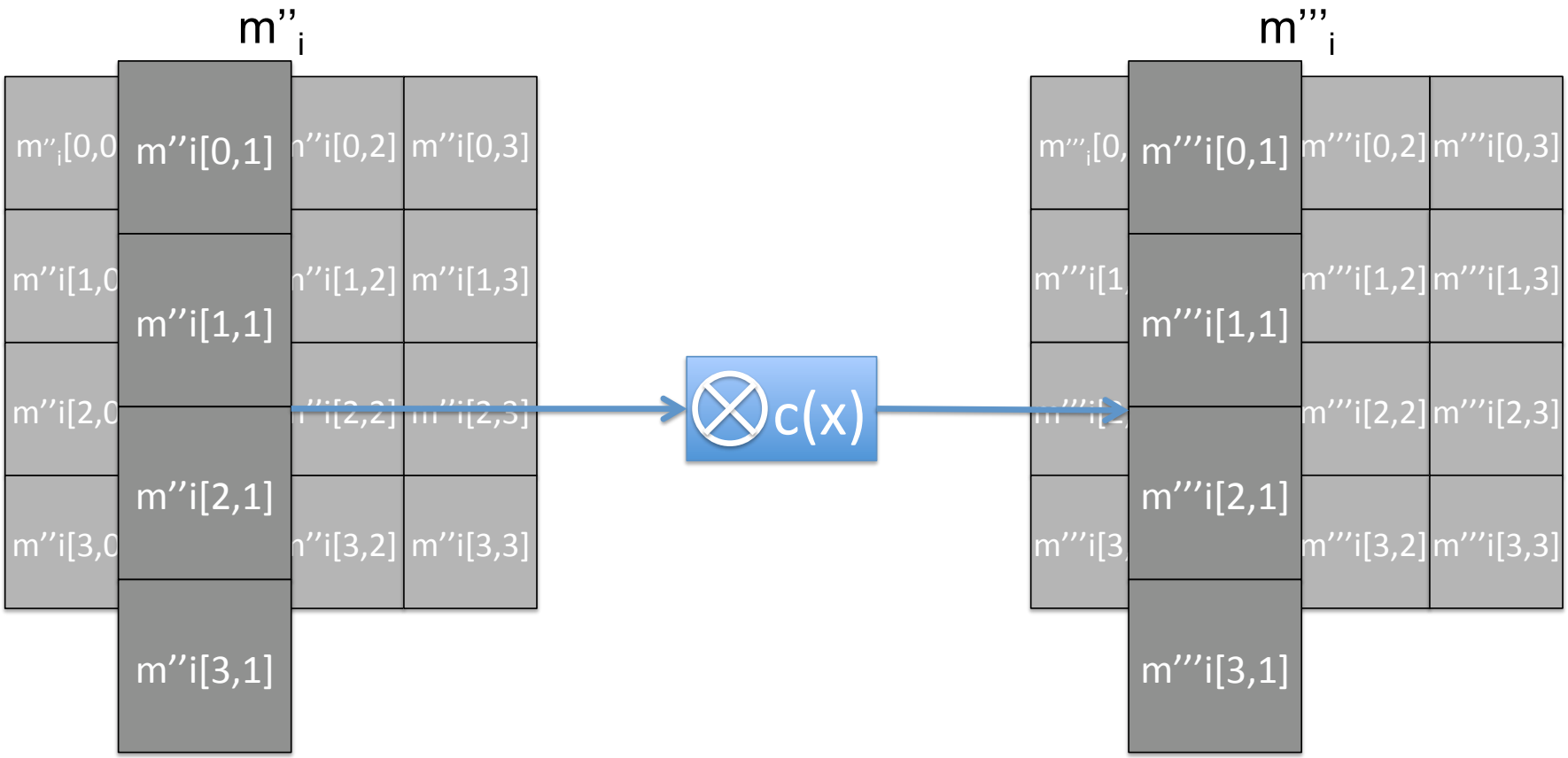
$m'_i[0,0]$	$m'_i[0,1]$	$m'_i[0,2]$	$m'_i[0,3]$
$m'_i[1,0]$	$m'_i[1,1]$	$m'_i[1,2]$	$m'_i[1,3]$
$m'_i[2,0]$	$m'_i[2,1]$	$m'_i[2,2]$	$m'_i[2,3]$
$m'_i[3,0]$	$m'_i[3,1]$	$m'_i[3,2]$	$m'_i[3,3]$



m''_i

$m'_i[0,0]$	$m'_i[0,1]$	$m'_i[0,2]$	$m'_i[0,3]$
$m'_i[1,1]$	$m'_i[1,2]$	$m'_i[1,3]$	$m'_i[1,0]$
$m'_i[2,2]$	$m'_i[2,3]$	$m'_i[2,0]$	$m'_i[2,1]$
$m'_i[3,3]$	$m'_i[3,0]$	$m'_i[3,1]$	$m'_i[3,2]$

AES: MixColumns



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

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

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

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 1111**1000 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**

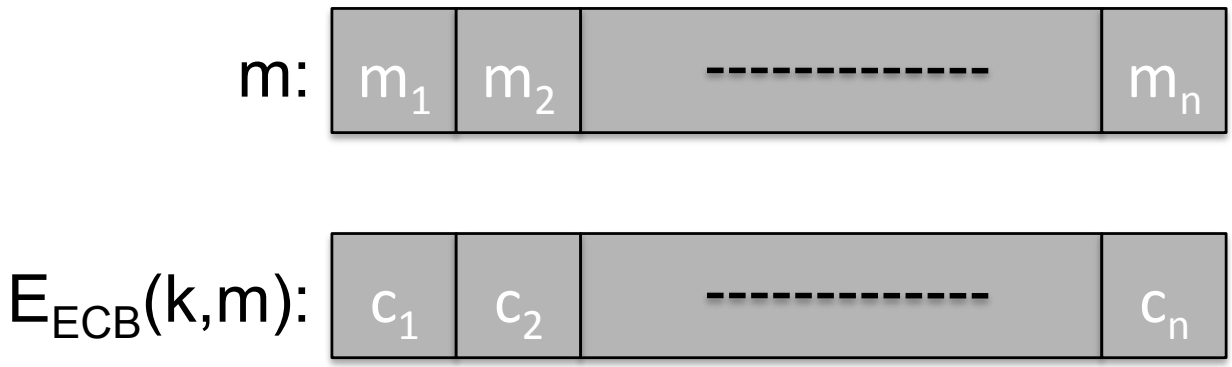
Electronic Code Book (ECB) mode

(E, D) a block cipher.

To encrypt message M under key K using ECB mode:

- M is padded:
  $\Rightarrow \; M' = M||P$ such that $|M'| = m \times \ell$
- 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 weak to frequency analysis!

Weakness of ECB in pictures

Original image

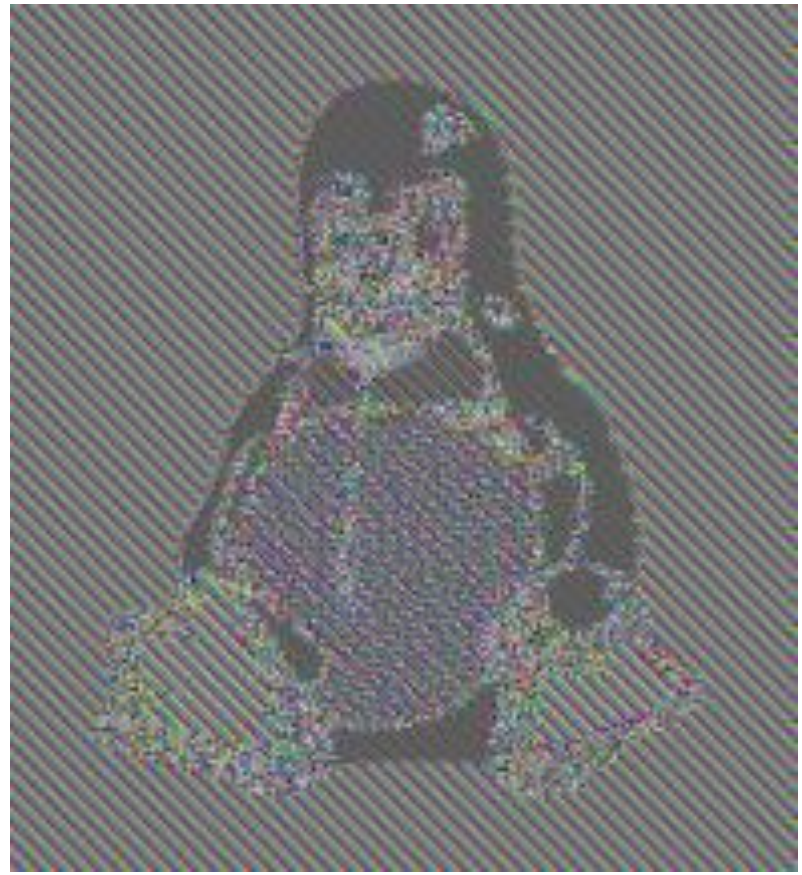
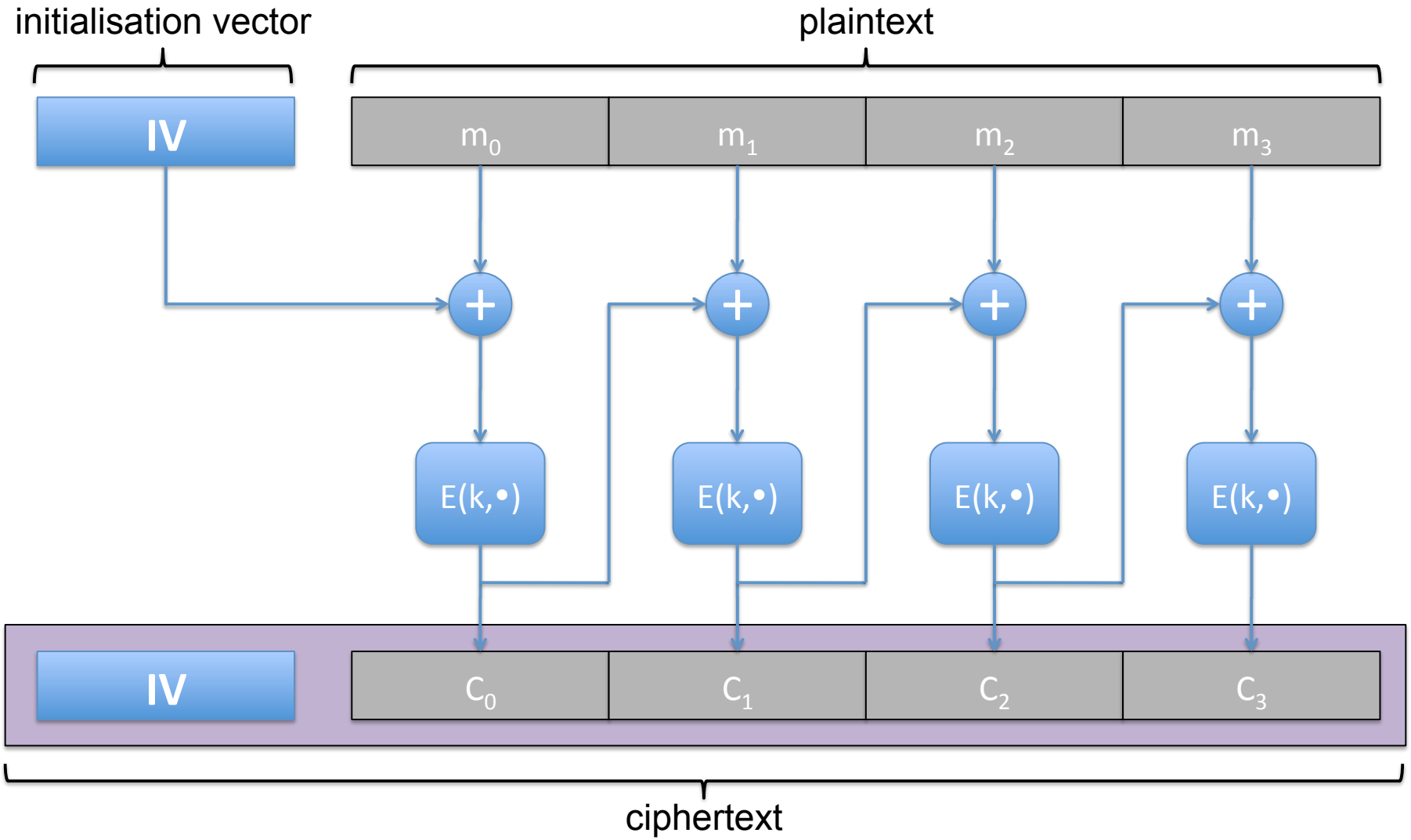


Image encrypted using ECB mode

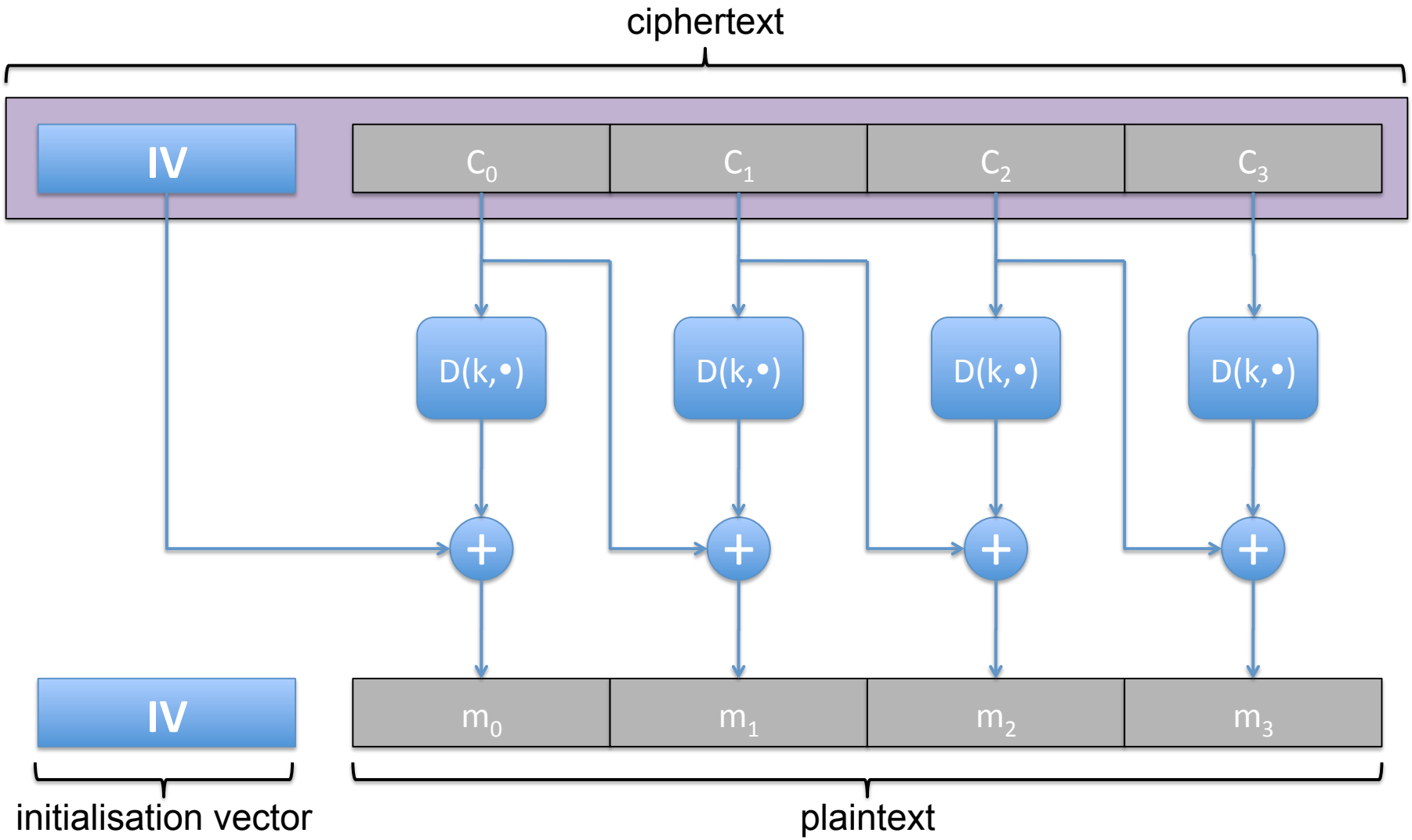
Cipher-block chaining (CBC) mode: encryption

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



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

Cipher-block chaining (CBC) mode: decryption



Sony PlayStation

- Prevent games being copied
- CD & full disk encryption
- Users can read and write on dedicated areas of disk
- Games loaded in read only area of disk
- With CBC encryption need to encrypt/decrypt whole disk to access a game
- Sony PS used ECB full-disk encryption

Sony PlayStation

- Prevent games being copied
- CD & full disk encryption
- Users can read and write on dedicated areas of disk
- Games loaded in read only area of disk
- With CBC encryption need to encrypt/decrypt whole disk to access a game
- Sony PS used ECB full-disk encryption
- Hardware controled user access to data

Sony PlayStation disk encryption attack

- Remove disk and make copy
- Put disk back in PlayStation
- Copy a file to the disk
- Remove disk and find area of disk that changed (that is the user encrypted file)
- Copy target data to the user area
- Put disk back in PlayStation and ask for user data
- PlayStation decrypts the file and gives it to user

Sweet32: birthday attacks on 64-bit block ciphers in TLS and openVPN

InfoWorld FROM IDG

INSIDER Sign

Home > Security

New collision attacks against triple-DES, Blowfish break HTTPS sessions

MORE LIKE THIS

Google to shutter SSLv3, RC4 from SMTP servers, Gmail

Researchers devise new attack techniques against SSL

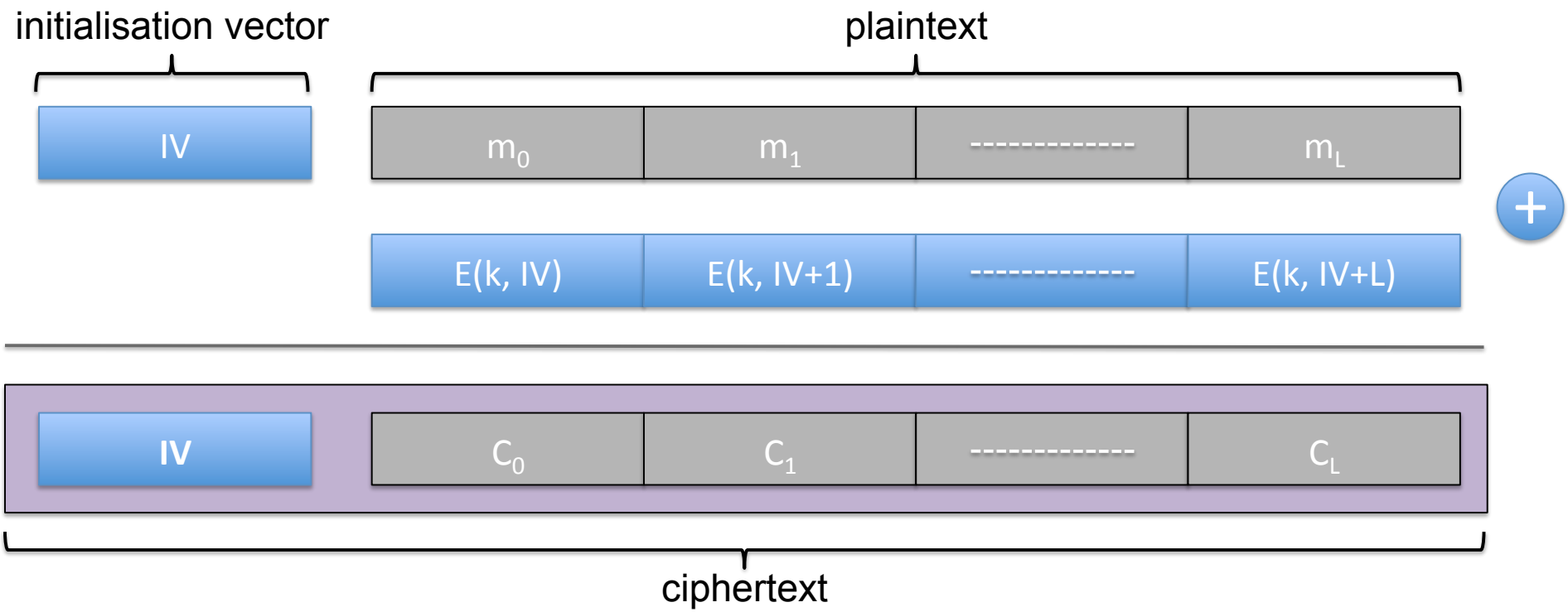
HTTP compression continues to put encrypted communications at risk

on IDG **Answers**

Can company see that I'm using their internet?

Counter (CTR) mode

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



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

Cryptography: cryptographic hash functions and MACs

Myrto Arapinis
School of Informatics
University of Edinburgh

January 30, 2019

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

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 end 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, SHA-256, Whirlpool, ...

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, ...

Collision resistance and the birthday attack

Theorem

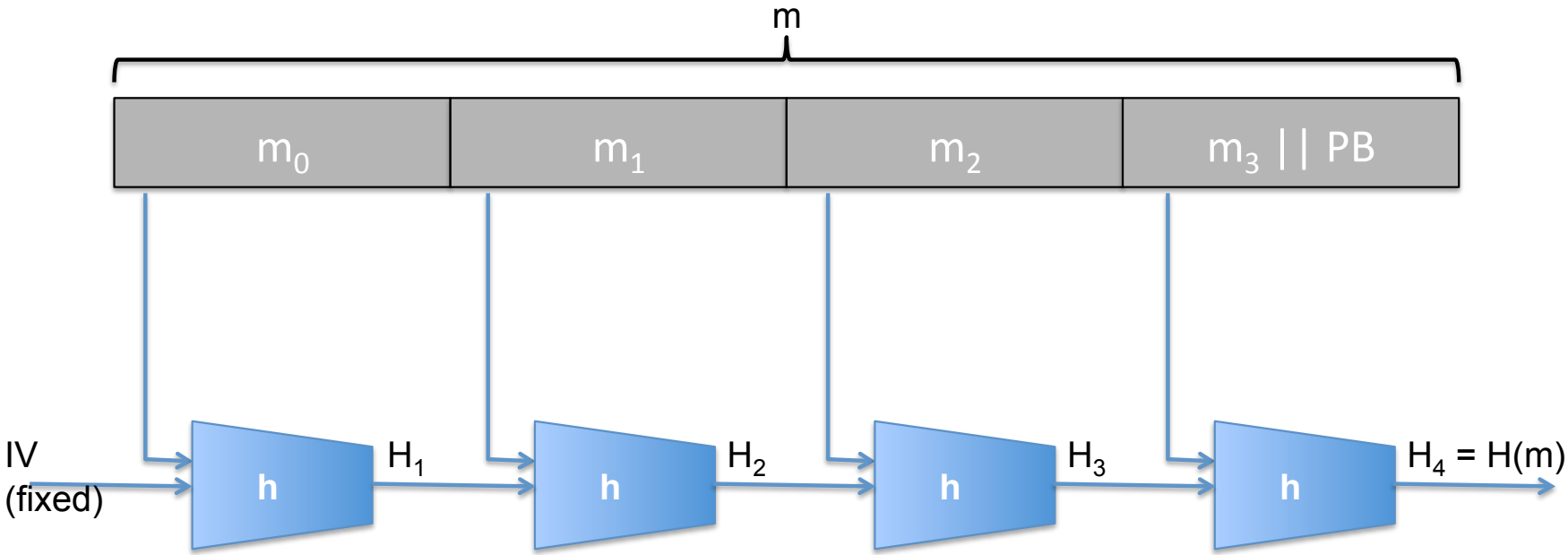
Let $H: \mathcal{M} \to \{0,1\}^n$ be a cryptographic hash function ($|\mathcal{M}| >> 2^n$)
Generic algorithm to find a collision in time $O(2^{n/2})$ hashes:

1. *Choose $2^{n/2}$ random messages in $\mathcal{M}$: $m_1, \ldots, m_{2^{n/2}}$*
2. *For $i = 1, \ldots, 2^{n/2}$ 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 $r_1, \ldots, r_n \in \{1, \ldots, N\}$ be independent variables.
For $n = 1.2 \times \sqrt{N}$, $Pr(\exists i \neq j.\ r_i = r_j) \geq \frac{1}{2}$
$\Rightarrow$ the expected number of iteration is 2
$\Rightarrow$ running time $O(2^{n/2})$

$\Rightarrow$ Cryptographic function used in new projects should have an output size $n \geq 256$!

The Merkle-Damgard construction



- Compression function: $h: \mathcal{T} \times \mathcal{X} \to \mathcal{T}$
- PB: $1000\ldots0||$mes-len (add extra block if needed)

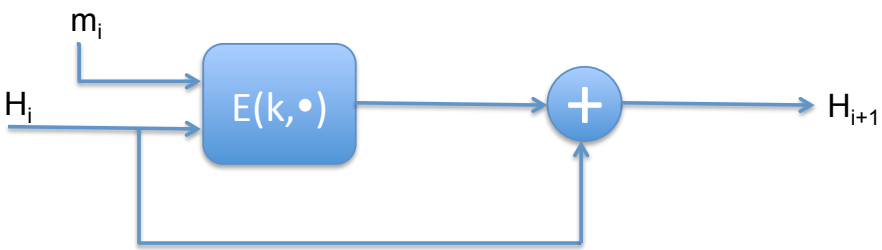
Theorem

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

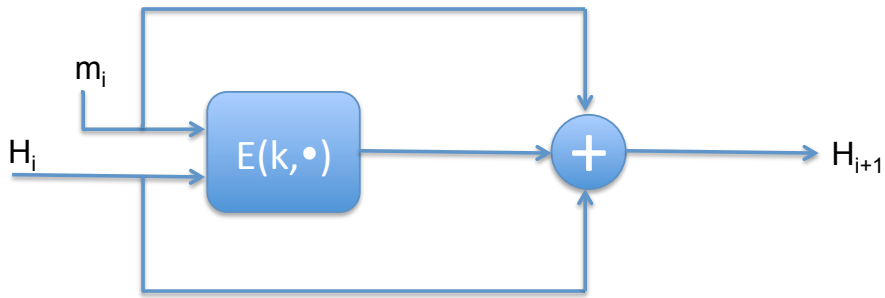
Example of MD constructions: MD5, SHA-1, SHA-2, ...

Compression functions from block ciphers

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



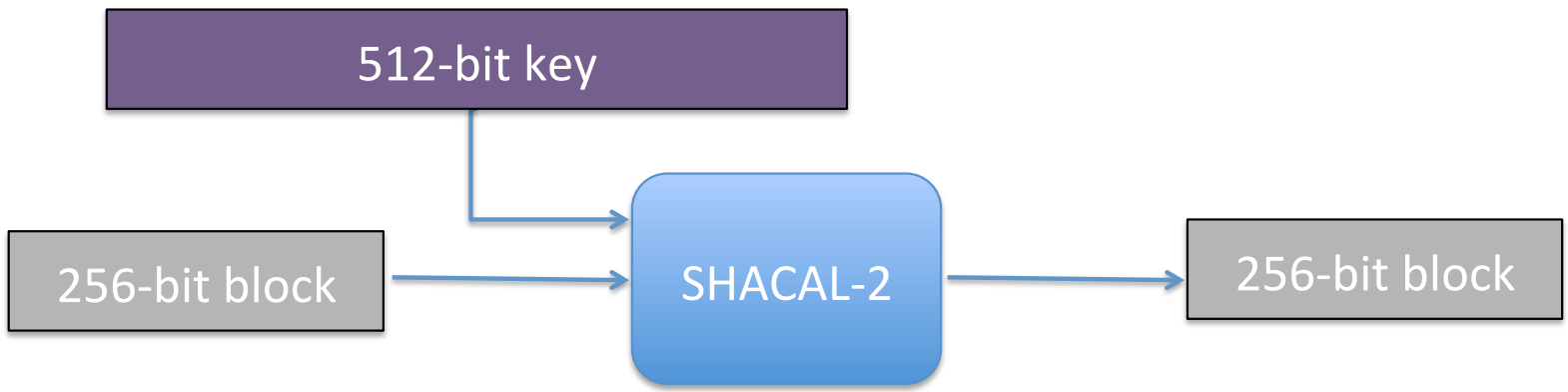
Davies-Meyer



Miyaguchi-Preneel

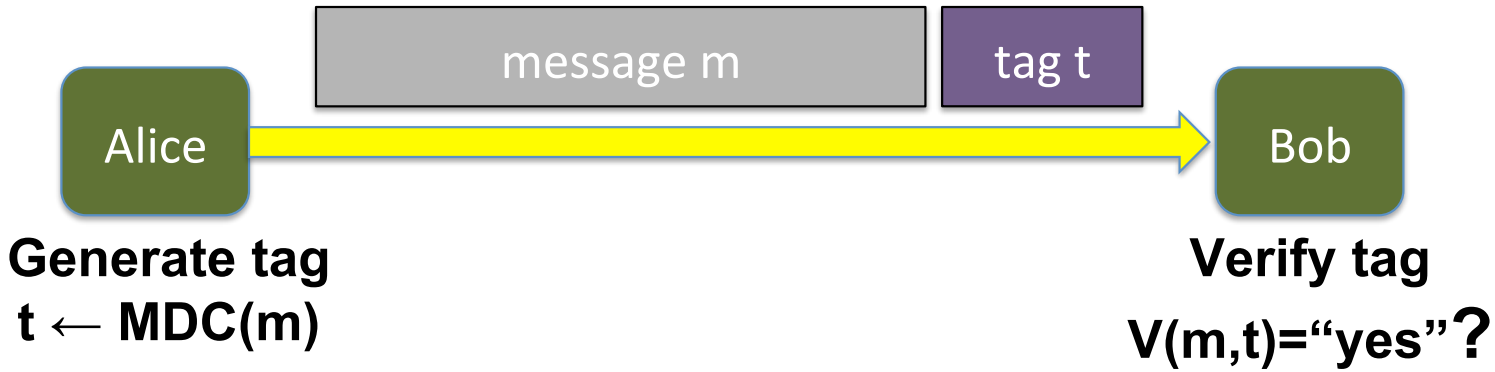
Example of cryptographic hash function: SHA-256

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

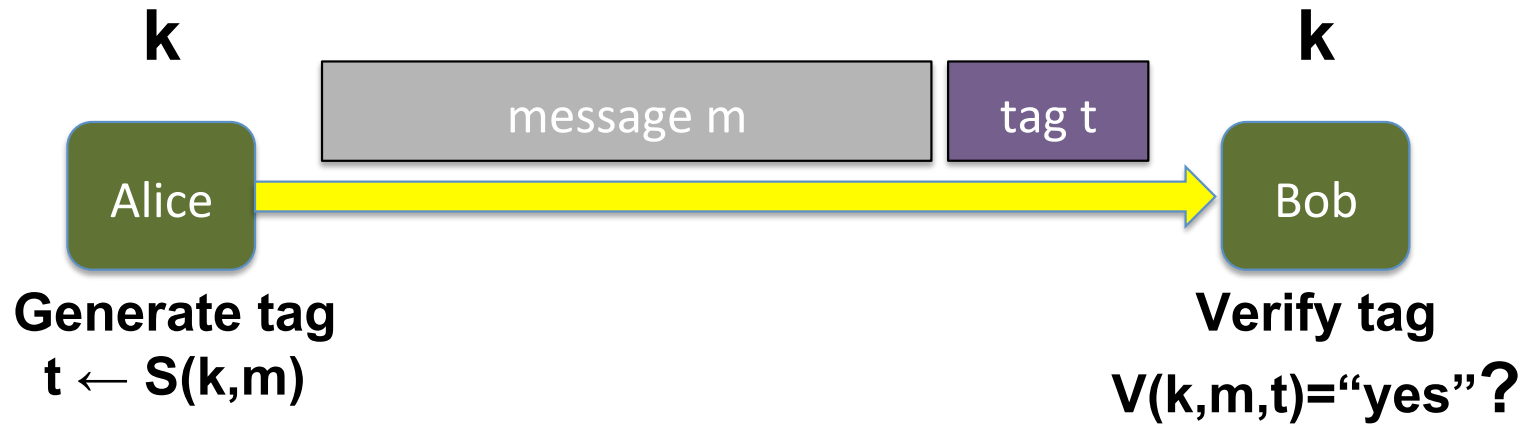


Message Authentication Codes (MACs)

Goal: message integrity



Goal: message integrity



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

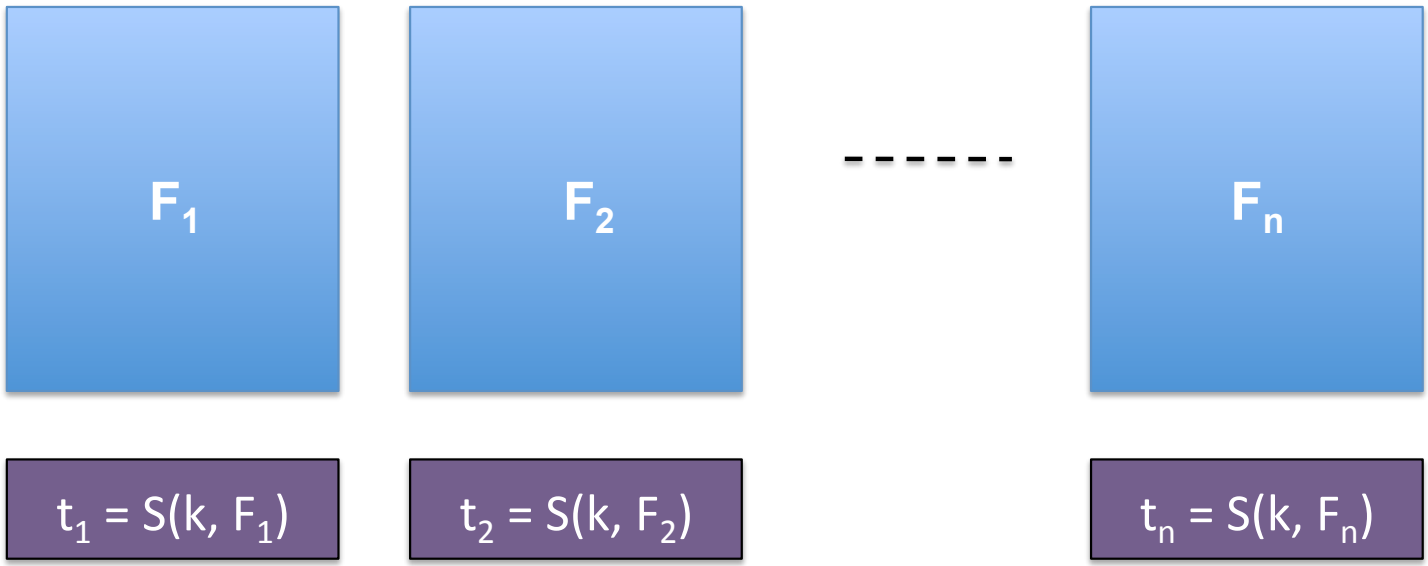
- $S : \mathcal{K} \times \mathcal{M} \to \mathcal{T}$
- $V : \mathcal{K} \times \mathcal{M} \times \mathcal{T} \to \{\top, \bot\}$
- Consistency: $V(k, m, S(k, m)) = T$

and such that

- 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

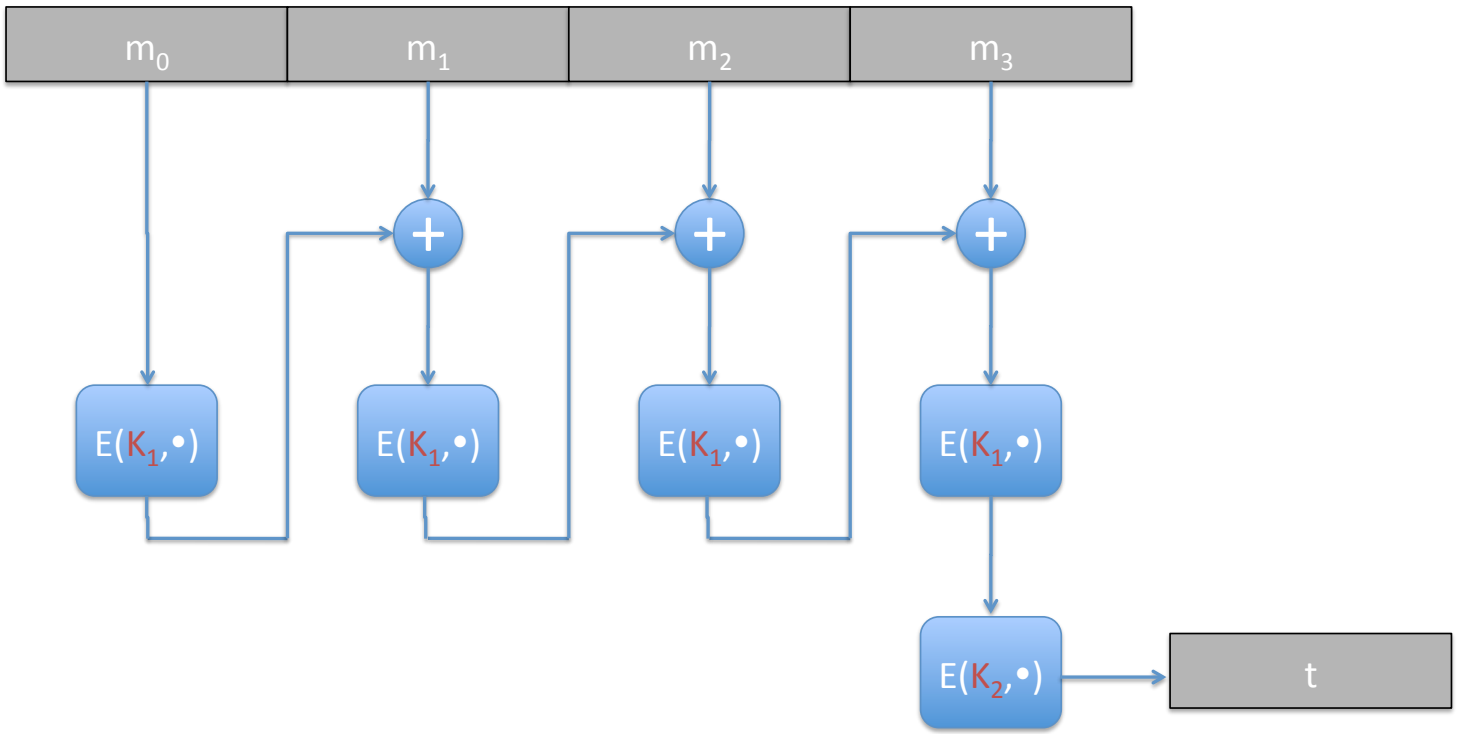
Let (E, D) be a block cipher. We build a MAC (S, V) using (E, D) as follows:

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

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

Our goal now: construct MACs for long messages

ECBC-MAC

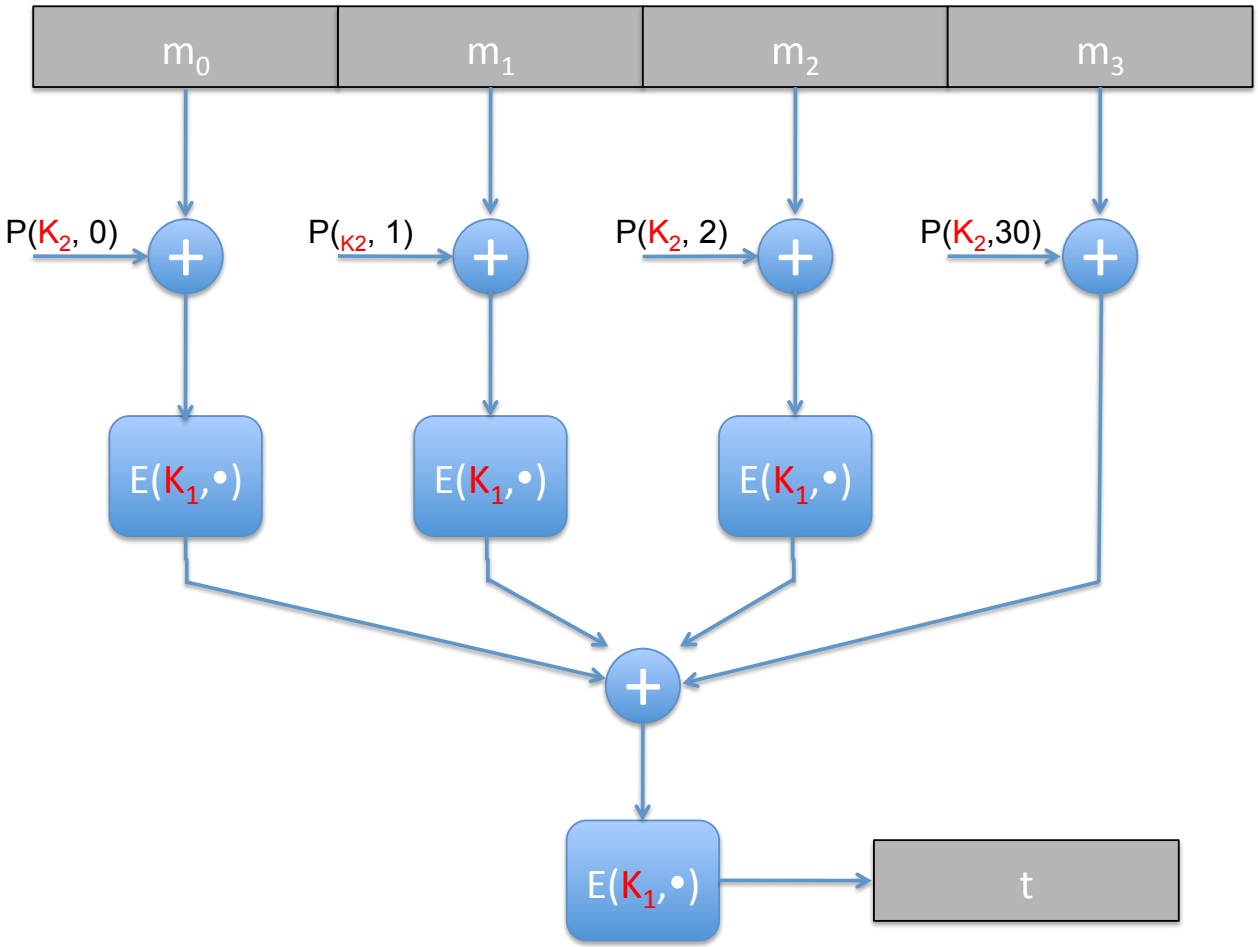


- $E : \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ a block cipher
- $ECBC\text{-}MAC : \mathcal{K}^2 \times \{0,1\}^* \to \{0,1\}^n$

$\to$ the last encryption is crucial to avoid forgeries!!

Ex: 802.11i uses AES based ECBC-MAC

PMAC



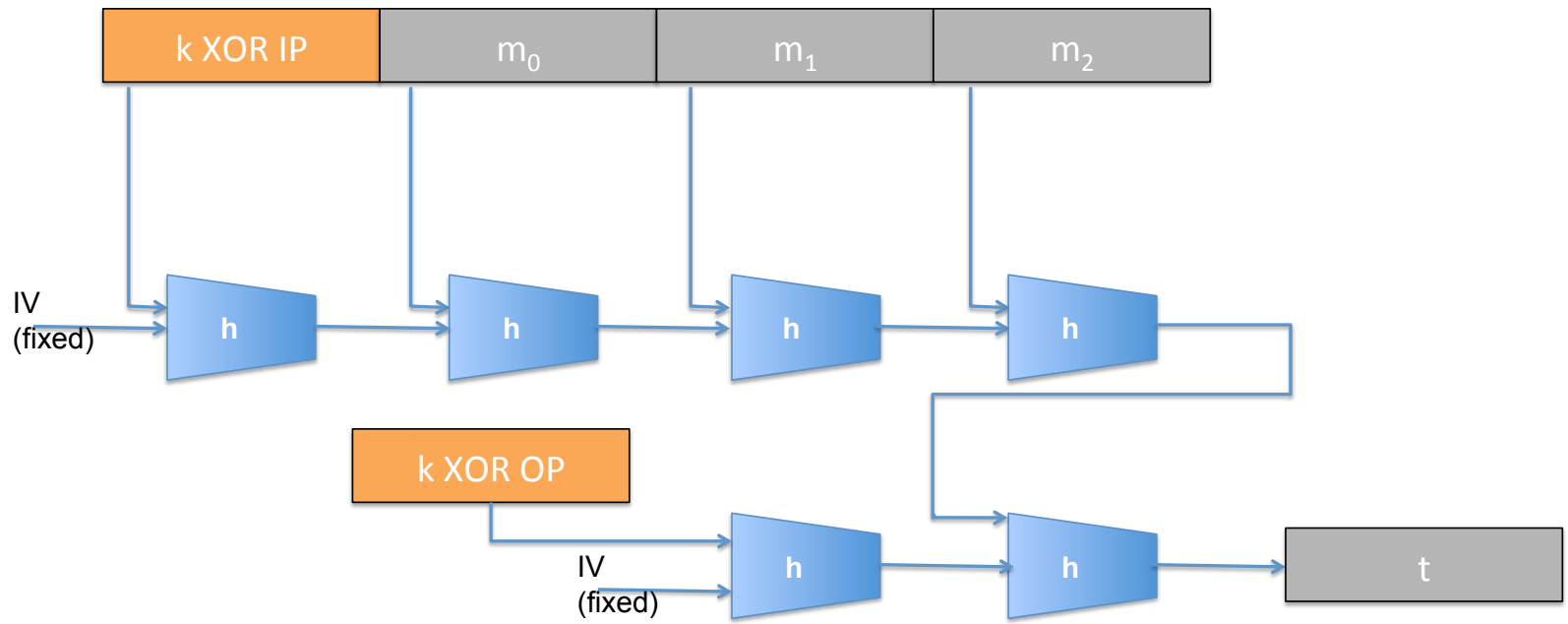
- $E : \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ a block cipher
- $P : \mathcal{K} \times \mathbb{N} \to \{0,1\}^n$ any easy to compute function
- $PMAC : \mathcal{K}^2 \times \{0,1\}^* \to \{0,1\}^n$

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



Ex: SSL, IPsec, SSH, ...

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
$\rightsquigarrow$ 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. $T \leftarrow \mathit{HMAC\text{-}SHA}(K_M, C)$
3. return $C||T$

Decryption

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

Do not:

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

Cryptography: asymmetric encryption

Myrto Arapinis
School of Informatics
University of Edinburgh

February 1, 2019

Introduction

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

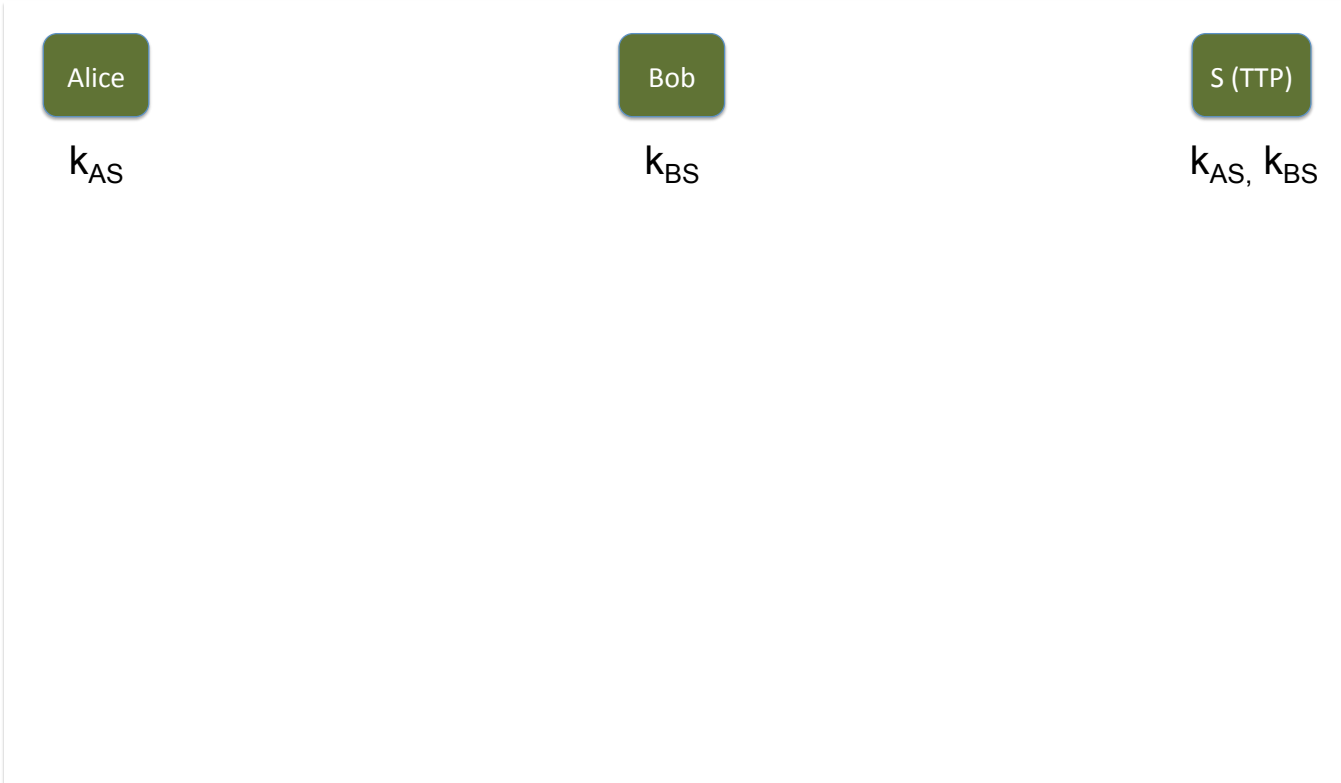
- One shared secret key per pair of users that want to communicate

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

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

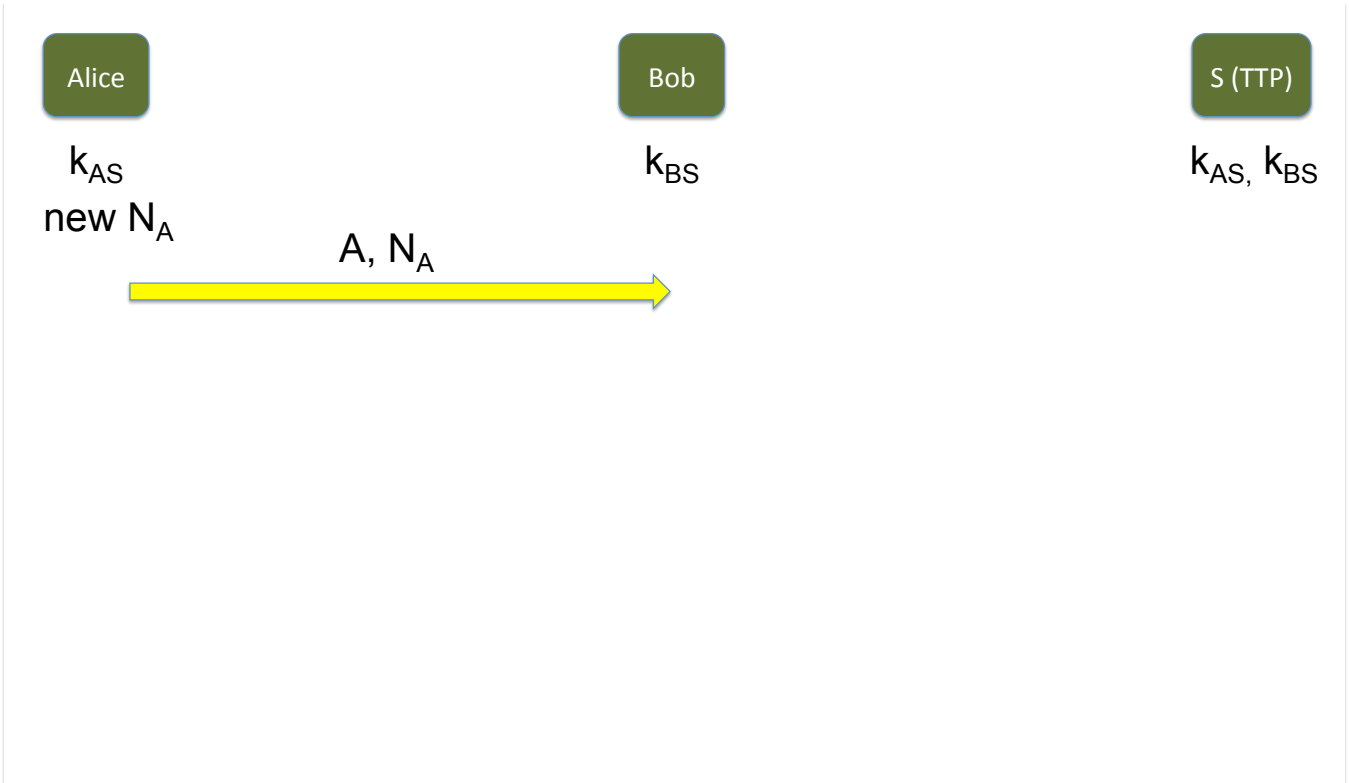
Online Trusted Third Party (TTP)

- 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
  ex: using Paulson's variant of the Yahalom protocol



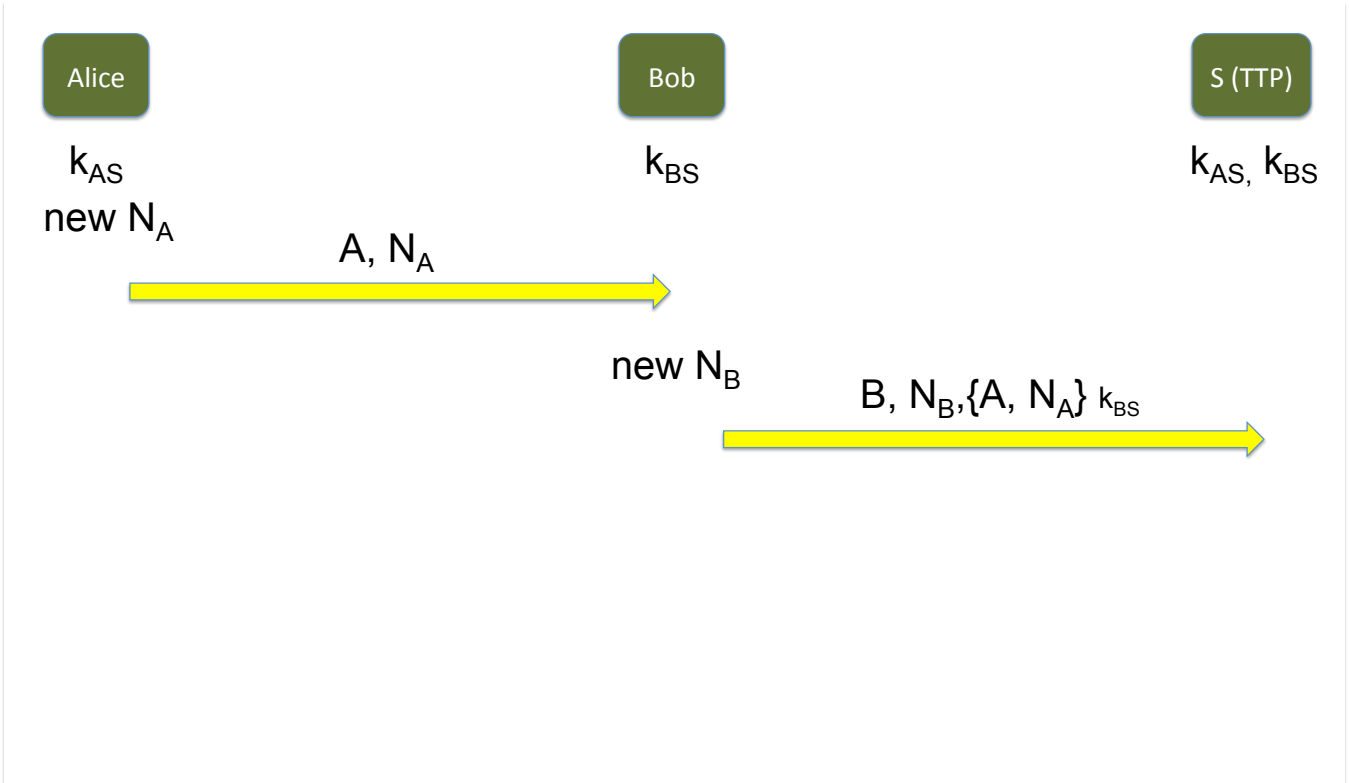
Online Trusted Third Party (TTP)

- 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
  ex: using Paulson's variant of the Yahalom protocol



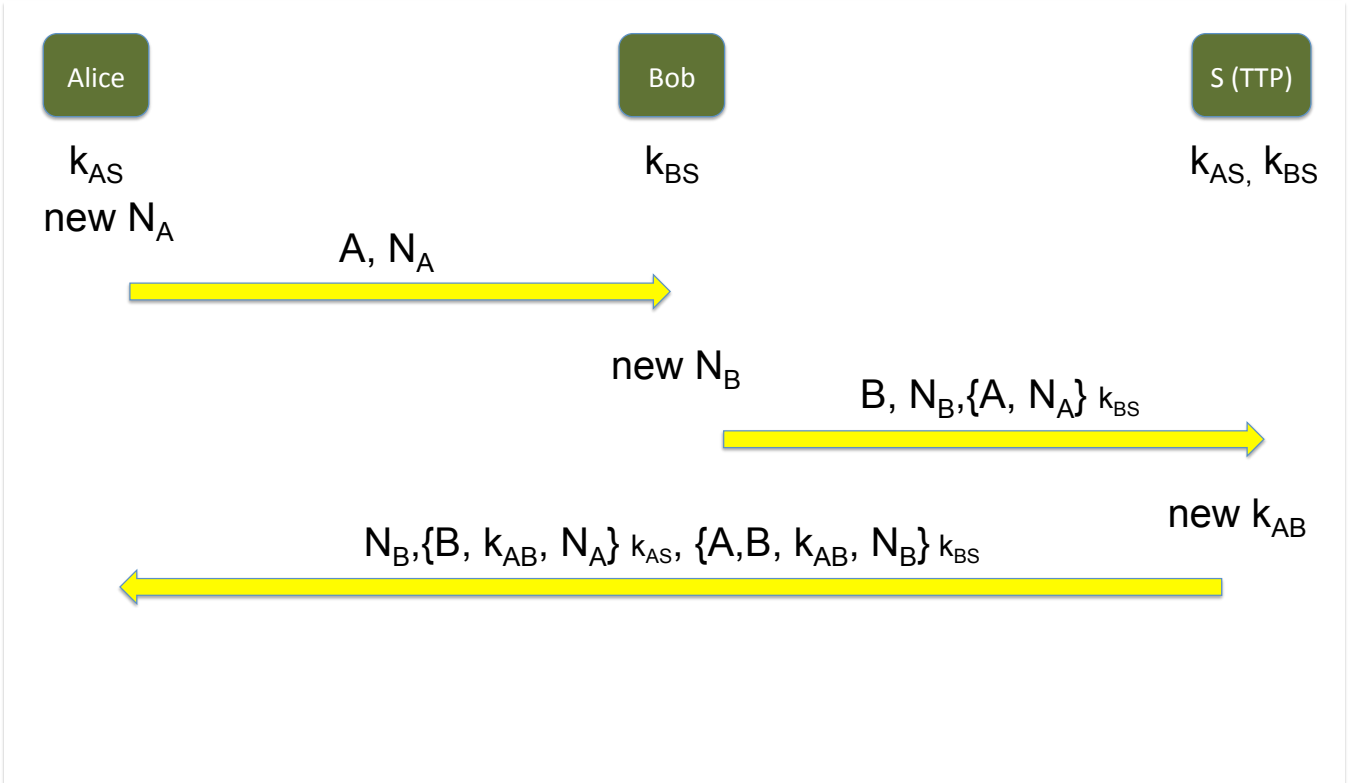
Online Trusted Third Party (TTP)

- 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
  ex: using Paulson's variant of the Yahalom protocol



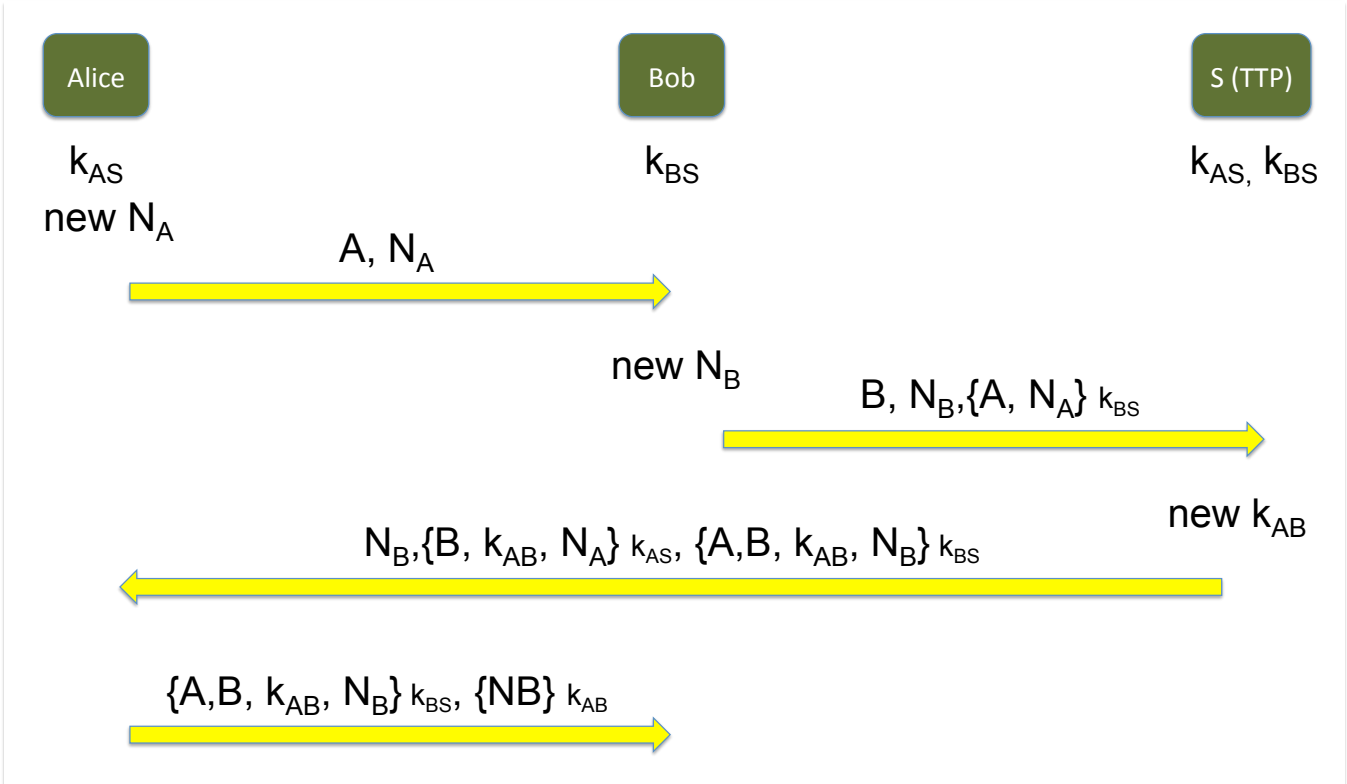
Online Trusted Third Party (TTP)

- 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
  ex: using Paulson's variant of the Yahalom protocol



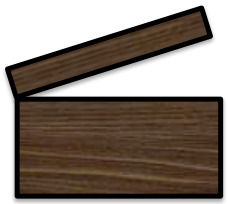
Online Trusted Third Party (TTP)

- 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
  ex: using Paulson's variant of the Yahalom protocol



Question: can we establish a shared secret key without a TTP?
Answer: Yes!

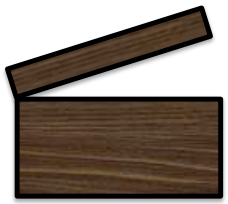
Public-key encryption in pictures



Alice

Bob

Public-key encryption in pictures

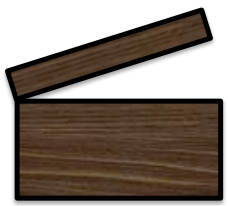


Alice

Bob

From Alice: I want to send you a secret

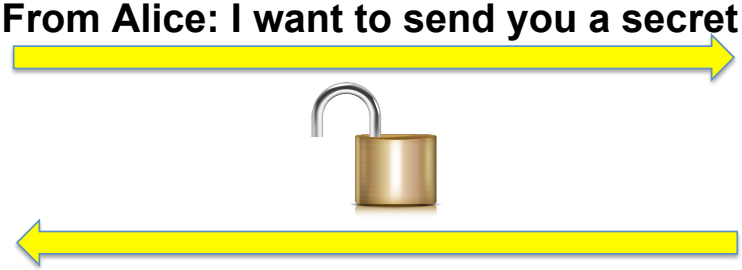
Public-key encryption in pictures



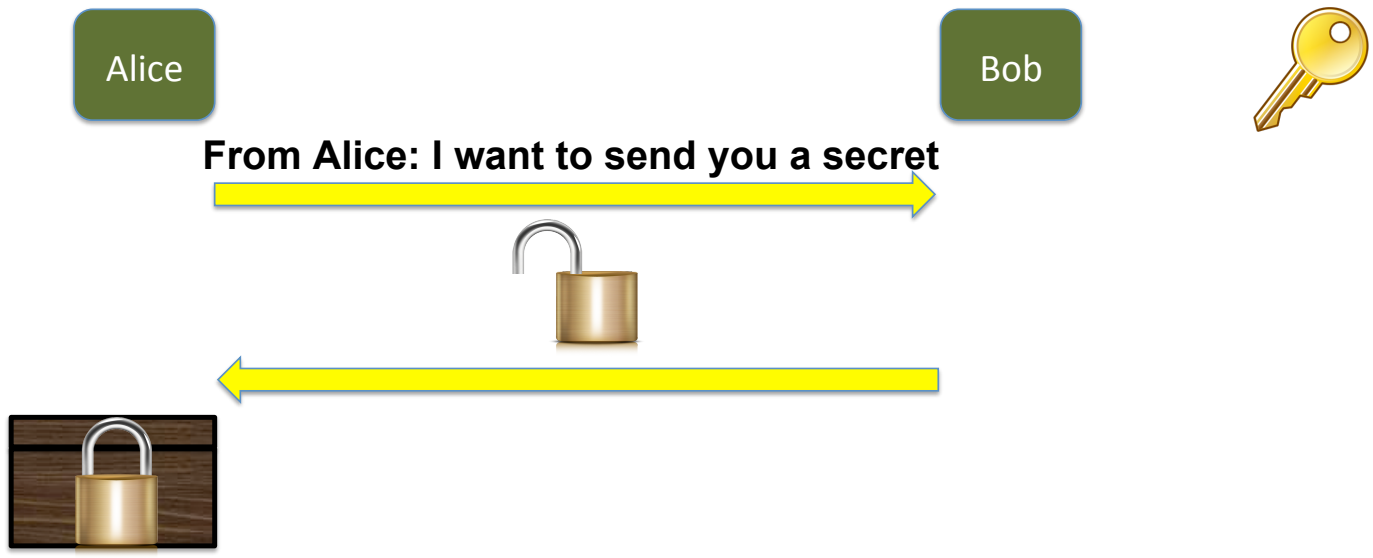
Alice

Bob

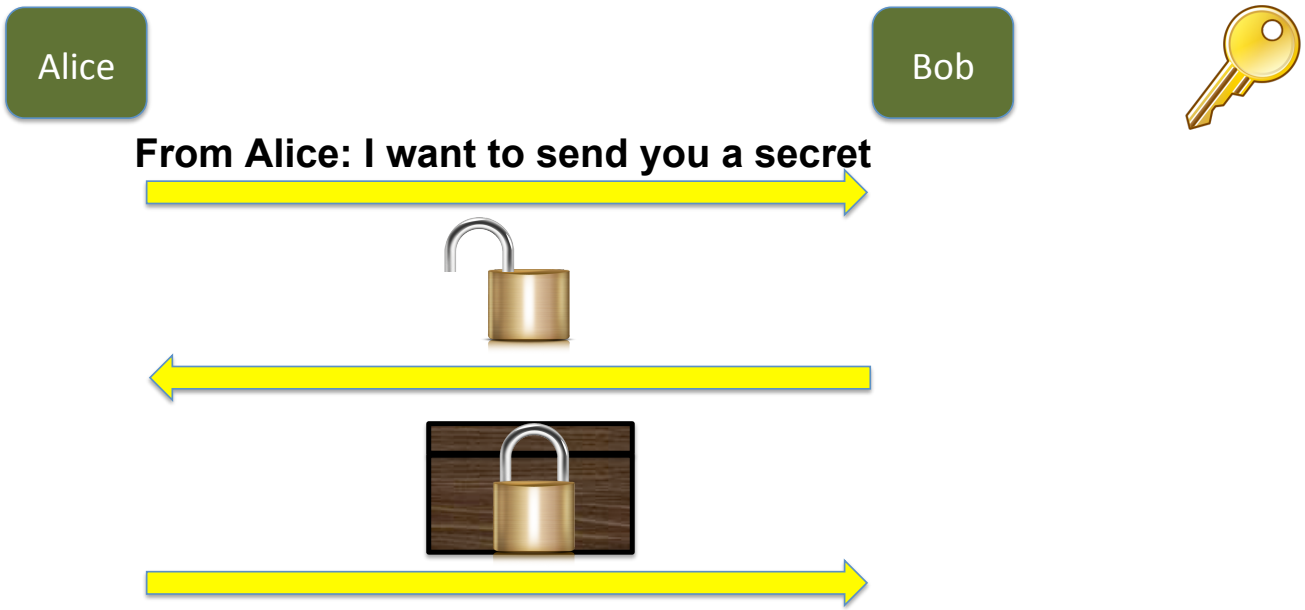
From Alice: I want to send you a secret



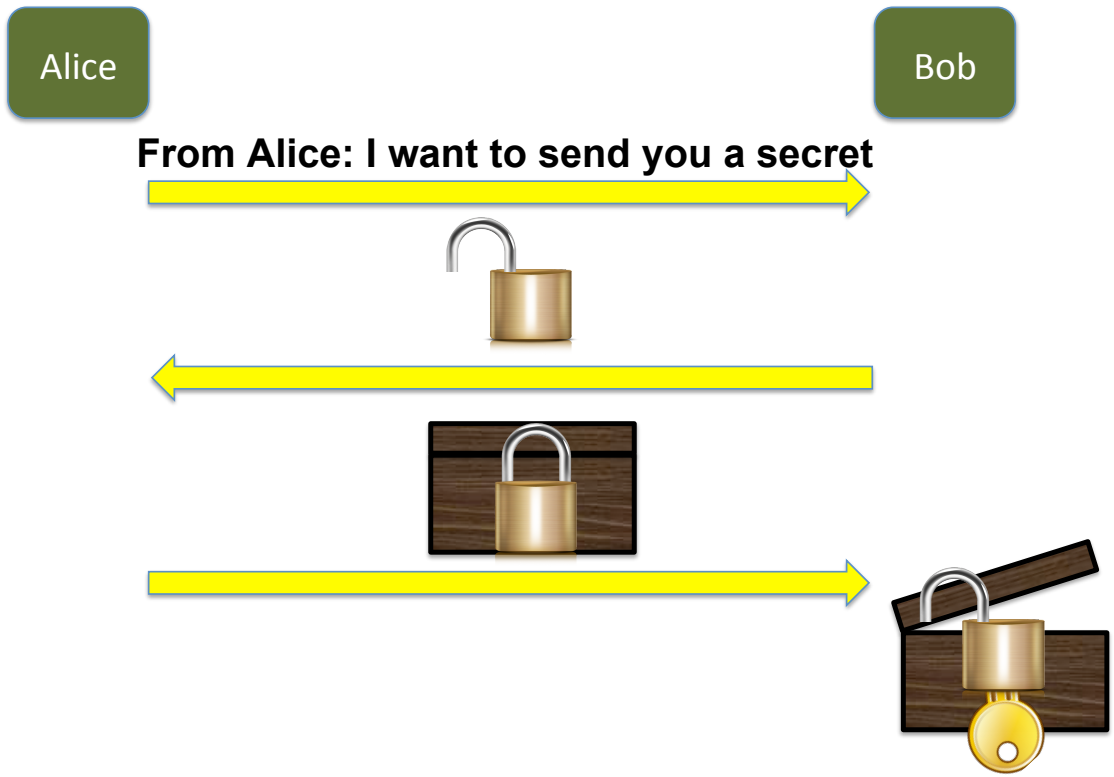
Public-key encryption in pictures



Public-key encryption in pictures

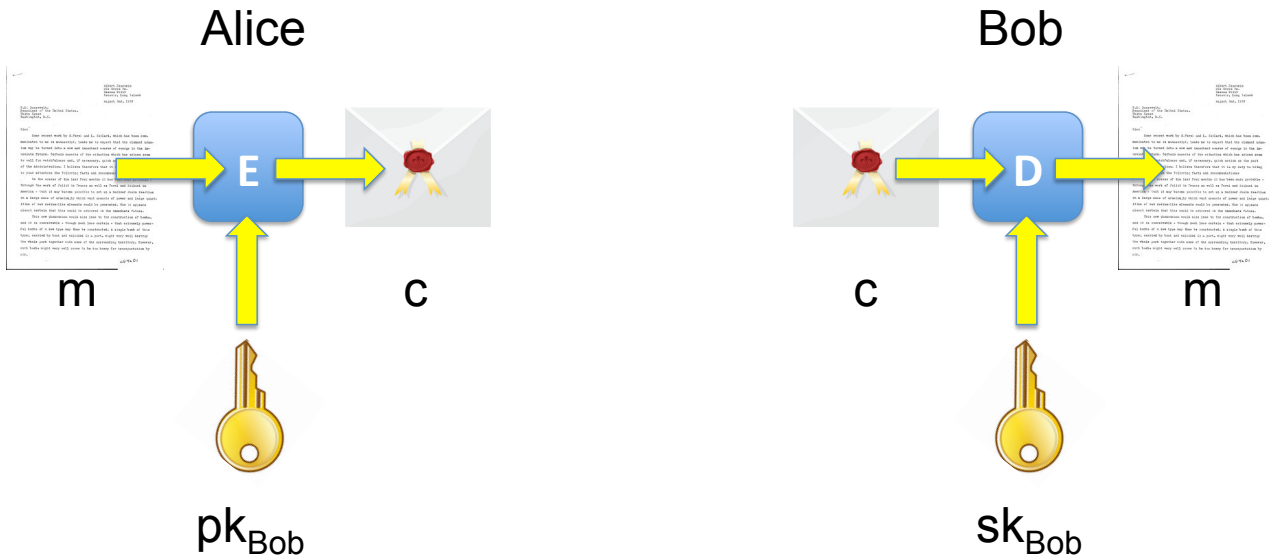


Public-key encryption in pictures



Public-key encryption

- key generation algorithm: $G : \rightarrow \mathcal{K} \times \mathcal{K}$
  encryption algorithm $E : \mathcal{K} \times \mathcal{M} \rightarrow \mathcal{C}$
  decryption algorithm $D : \mathcal{K} \times \mathcal{C} \rightarrow \mathcal{M}$
  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 $sk_{Bob} \neq pk_{Bob}$

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: $\phi(p \cdot q) = (p\text{-}1)(q\text{-}1)$

$\mathbb{Z}_n$

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

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

- 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 $\gcd(x, n) = 1$

$\mathbb{Z}_N^*$

- 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)

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

Theorem (Euler)

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

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

Intractable problems

- FACTORING:
  input: $n \in \mathbb{N}$
  output: $p_1, \ldots, p_m$ primes *st.* $n = p_1 \cdot \cdots \cdot p_m$
- RSAP
  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 LOG:
  input: prime p, generator g of $\mathbb{Z}_p^*$, $y \in \mathbb{Z}_p^*$
  output: x such that $y = g^x \pmod p$
- DHP:
  input: prime p, generator g of $\mathbb{Z}_p^*$, $g^a \pmod p$, $g^b \pmod p$
  output: $g^{ab} \pmod p$

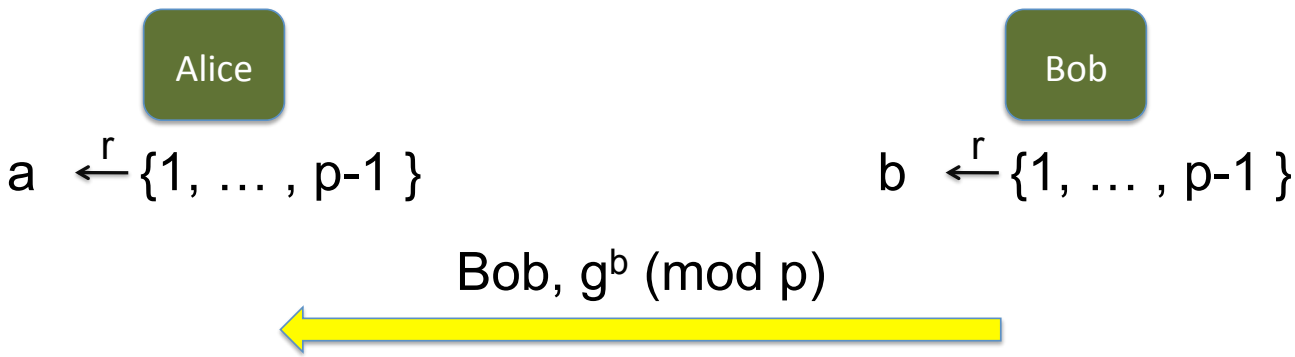
We can now go back and see how to establish a key without a TTP

The Diffie-Hellman (DH) protocol

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

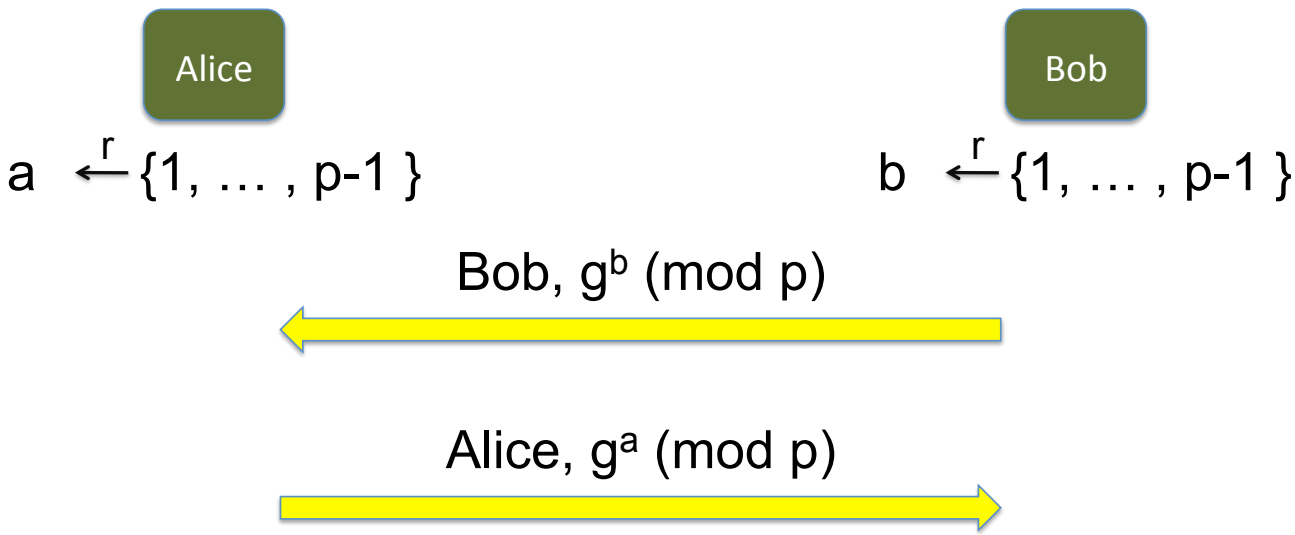
The Diffie-Hellman (DH) protocol

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



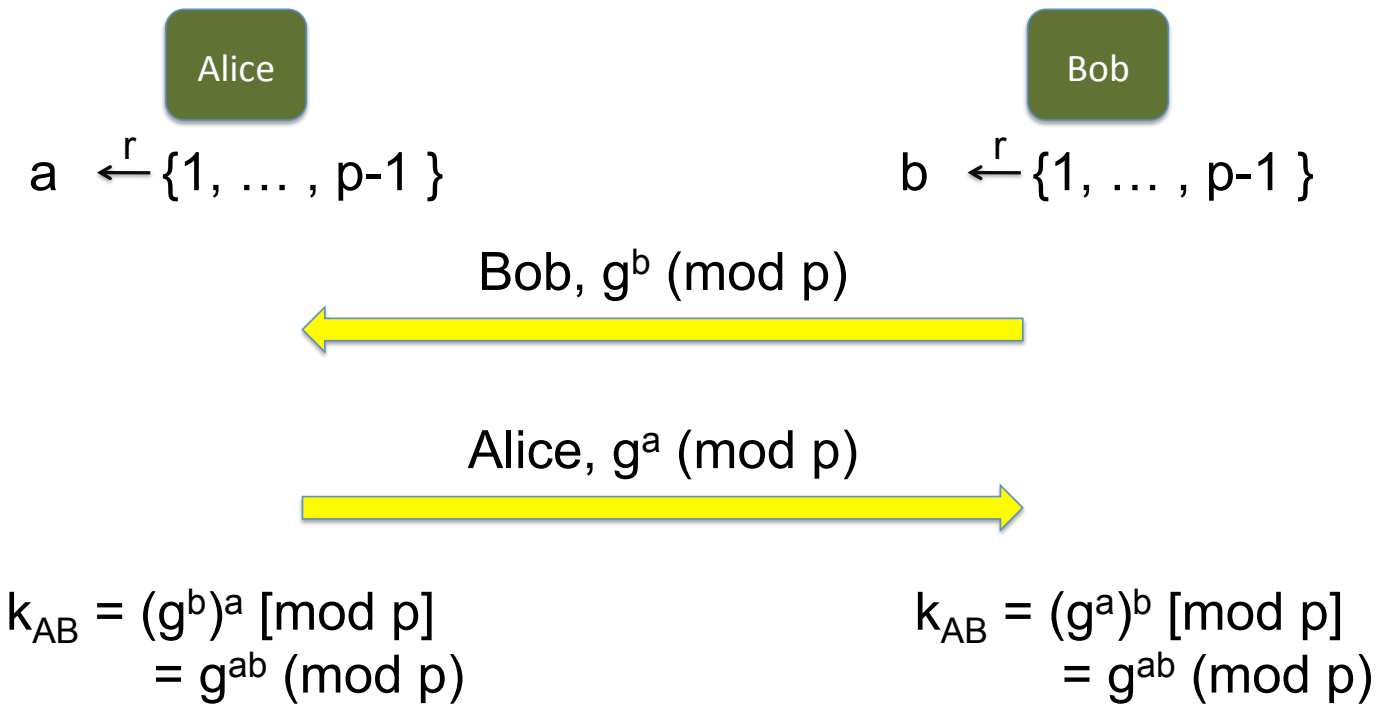
The Diffie-Hellman (DH) protocol

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

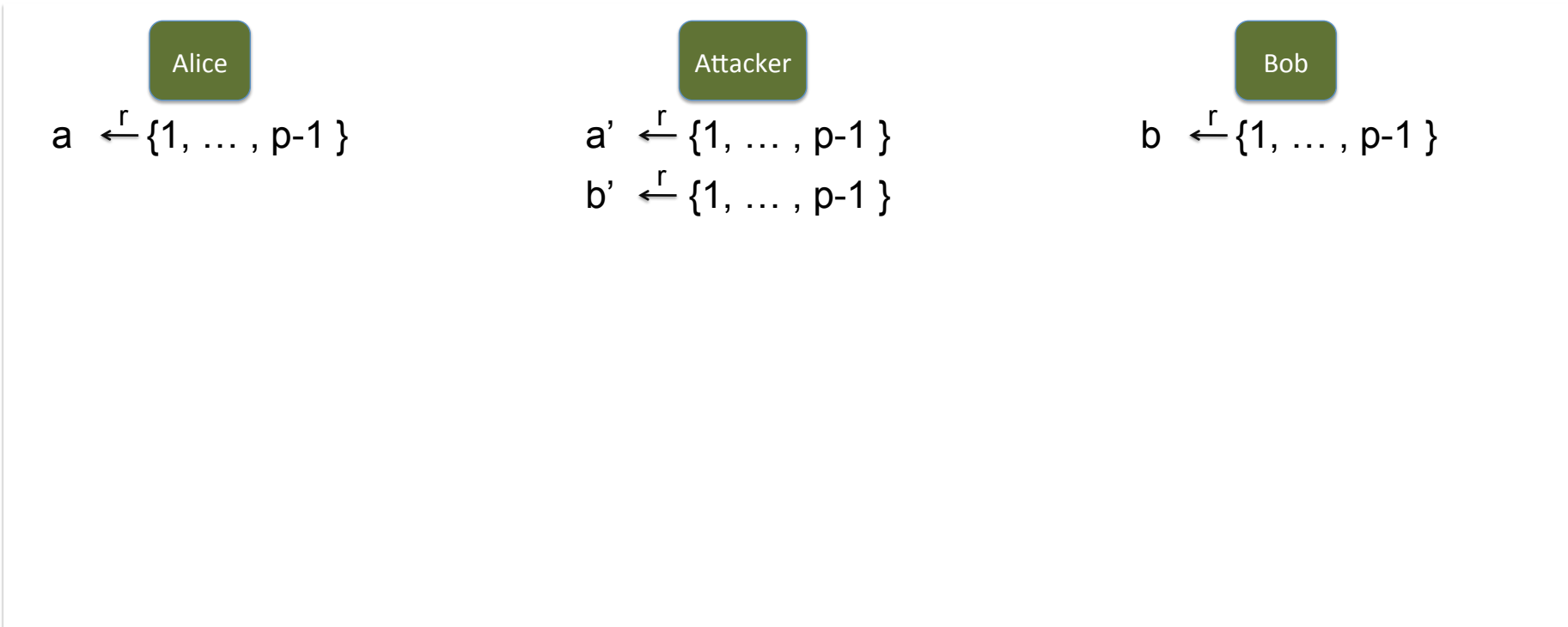


The Diffie-Hellman (DH) protocol

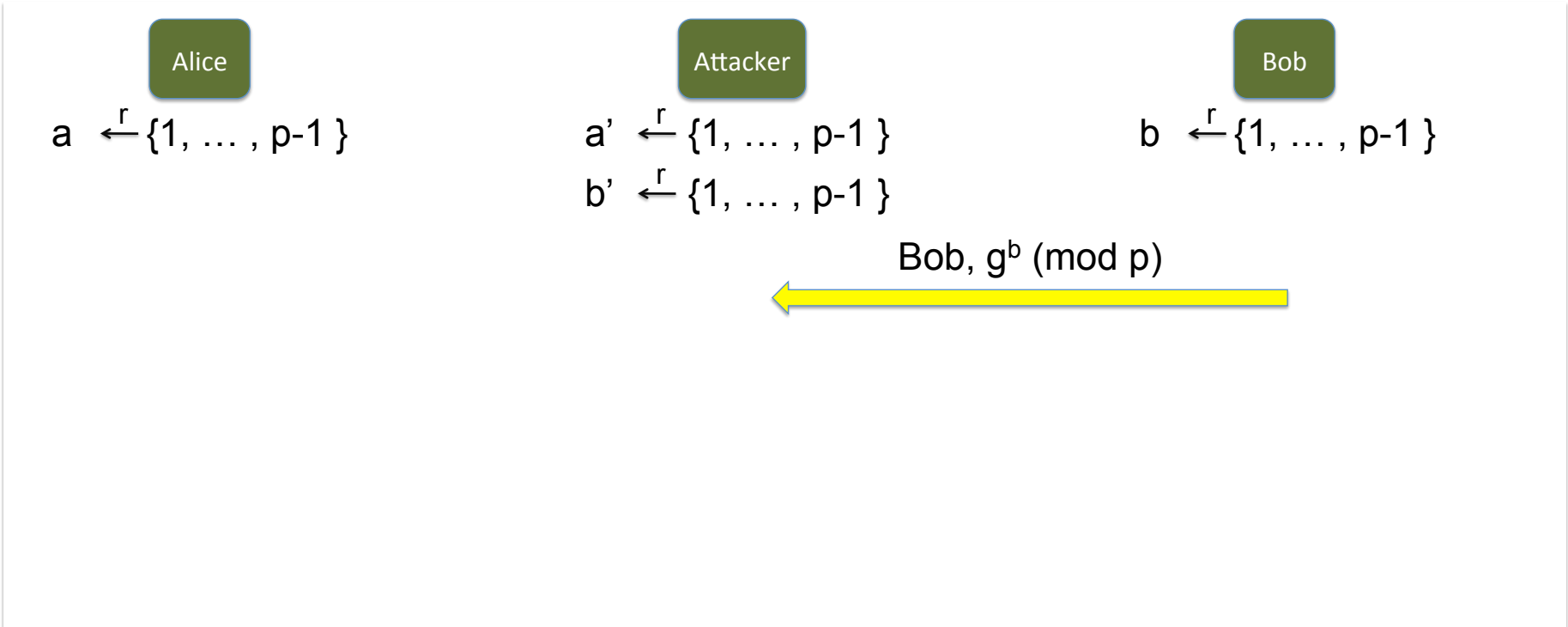
- Assumption: the DHP is hard in $\mathbb{Z}_p^*$
- Fix a very large prime p, and $g \in \{1, \ldots, p\text{-}1\}$



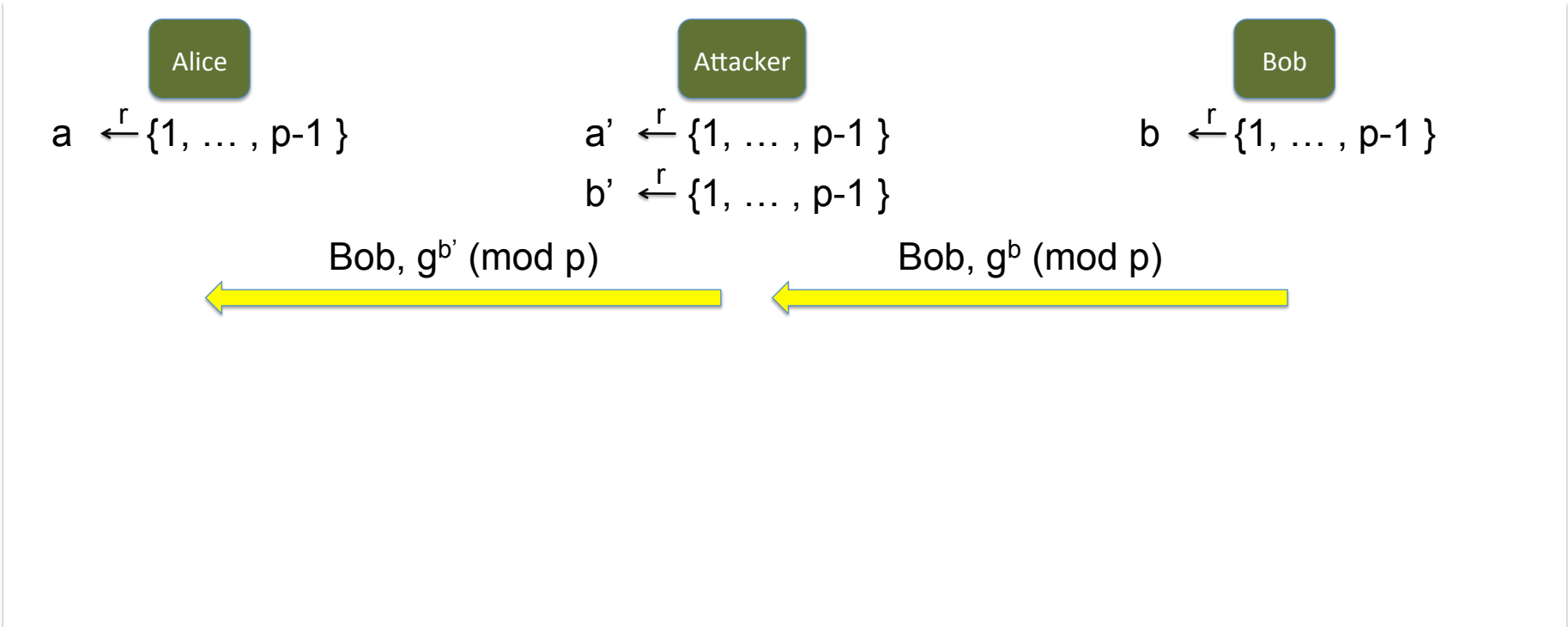
Man in the middle attack on DH



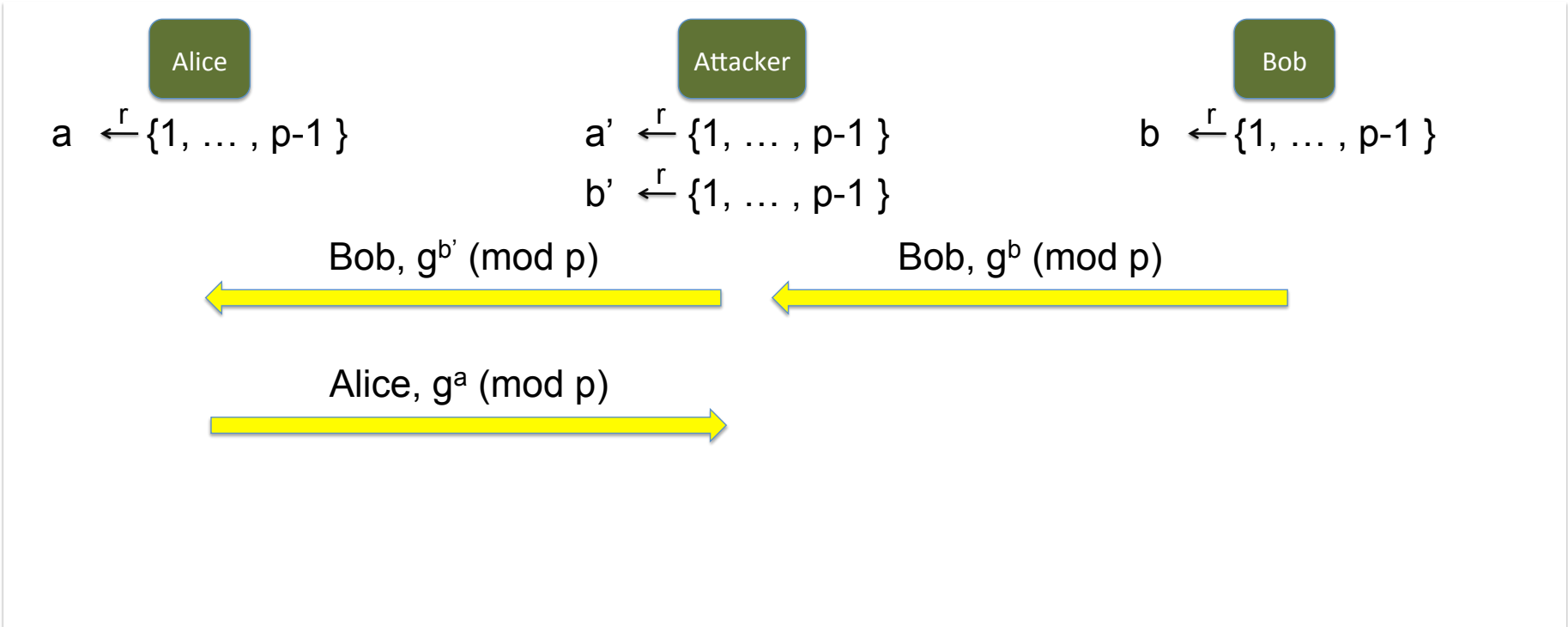
Man in the middle attack on DH



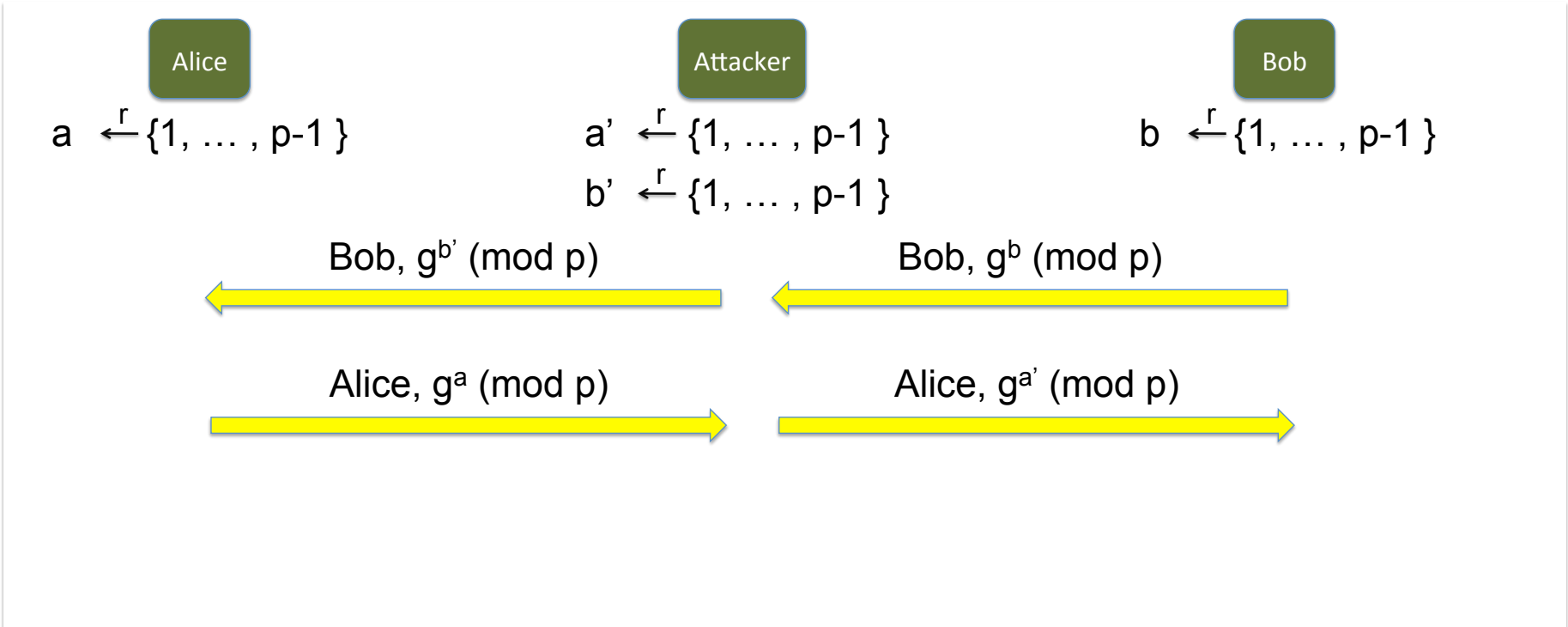
Man in the middle attack on DH



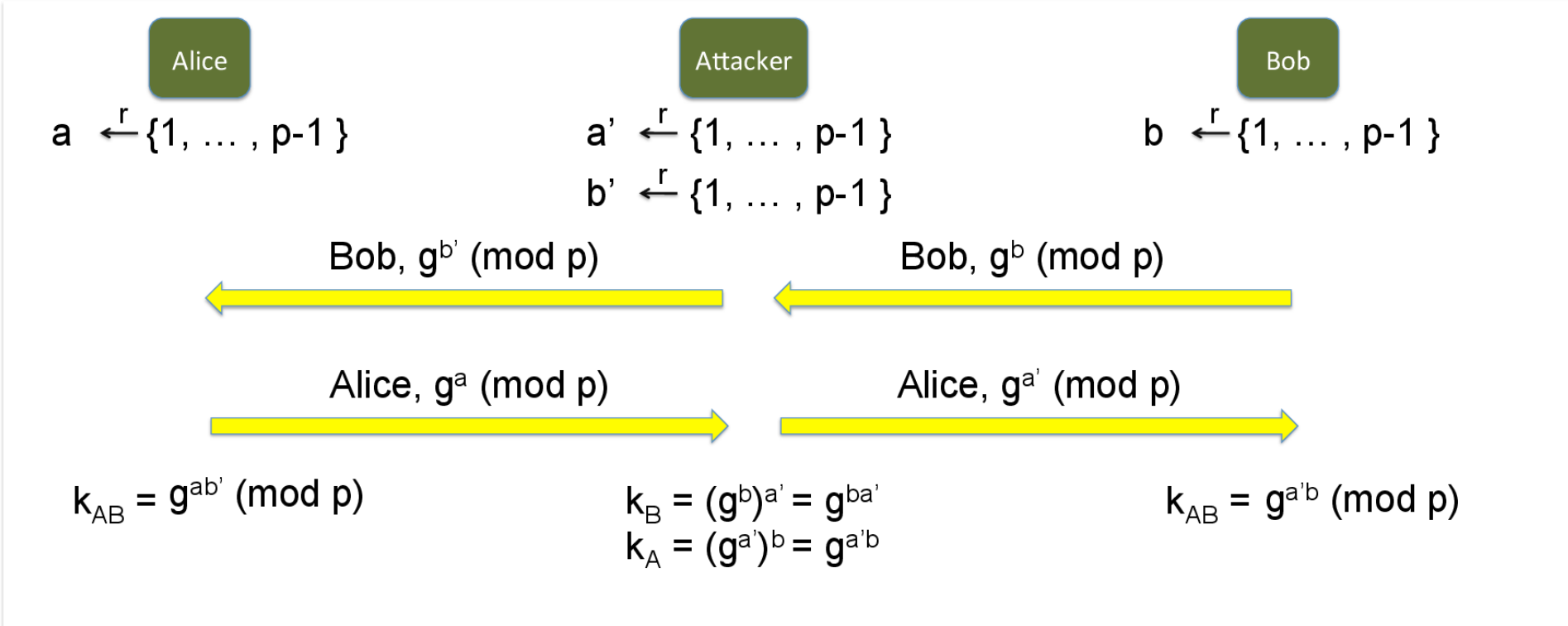
Man in the middle attack on DH



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*. Do not use raw *RSA*directly as an asymmetric cipher!

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

ISO standard

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

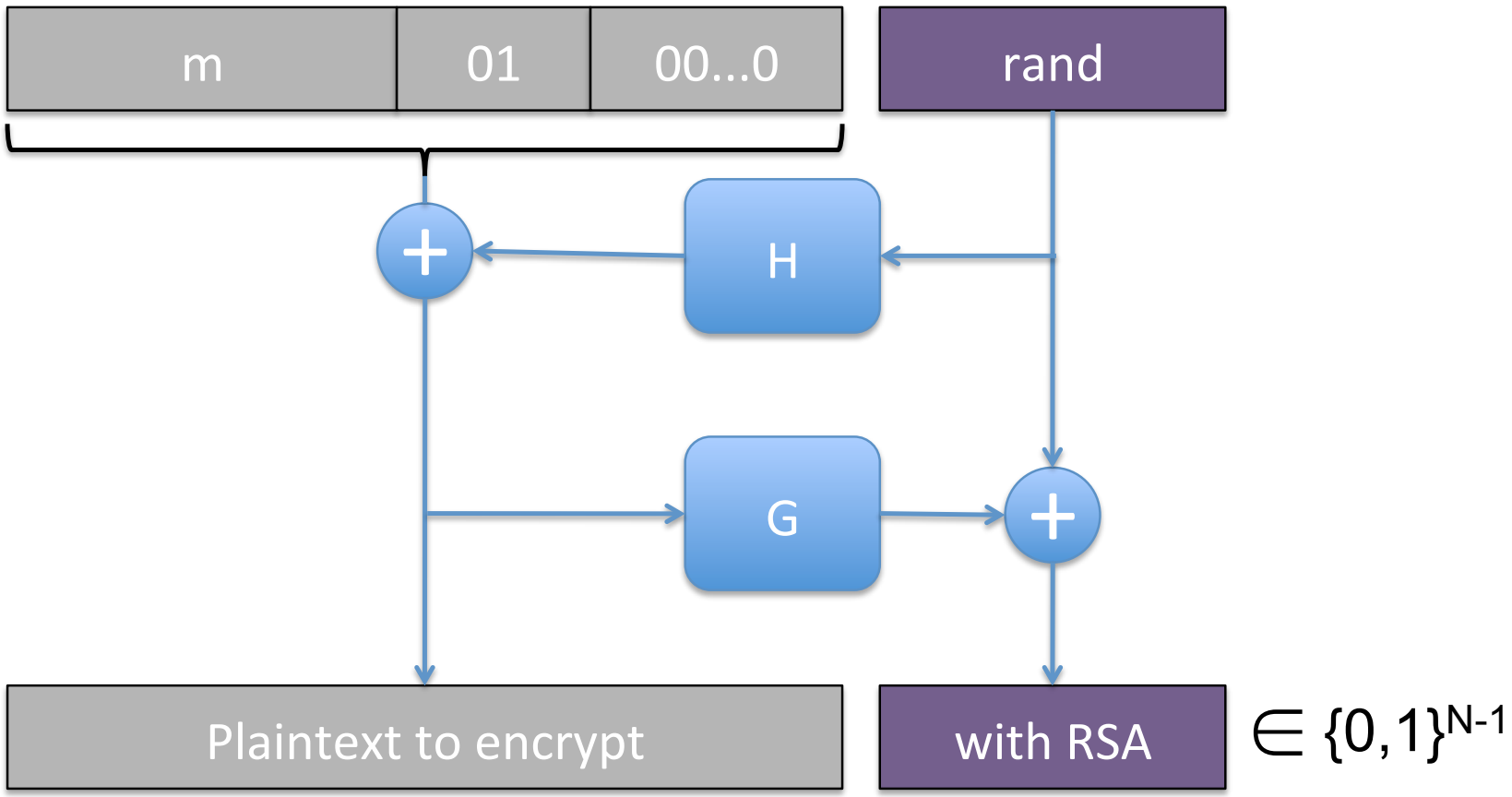
Let (E_s, D_s) be a symmetric encryption scheme over $(\mathcal{M}, \mathcal{C}, \mathcal{K})$
Let $H: \mathbb{Z}_N^* \to \mathcal{K}$

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

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

PKCS1 v2.0: *RSA*-OAEP

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



ElGamal (EG)

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

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

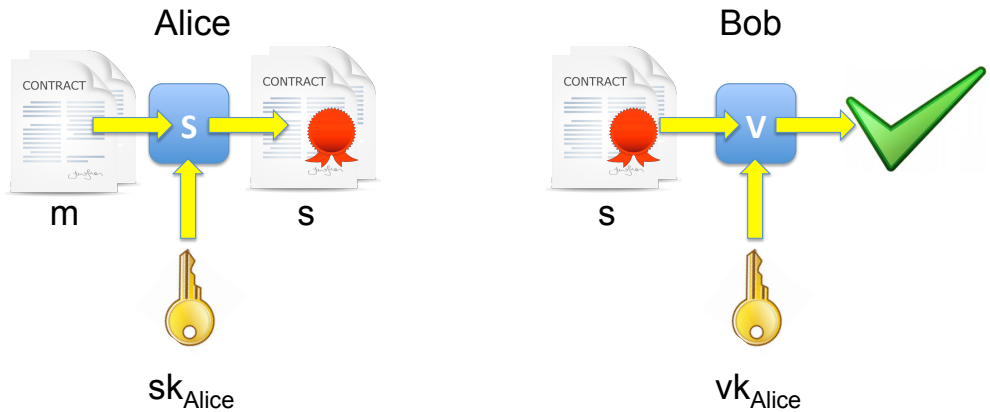
Cryptography: digital signatures

Myrto Arapinis
School of Informatics
University of Edinburgh

February 4, 2019

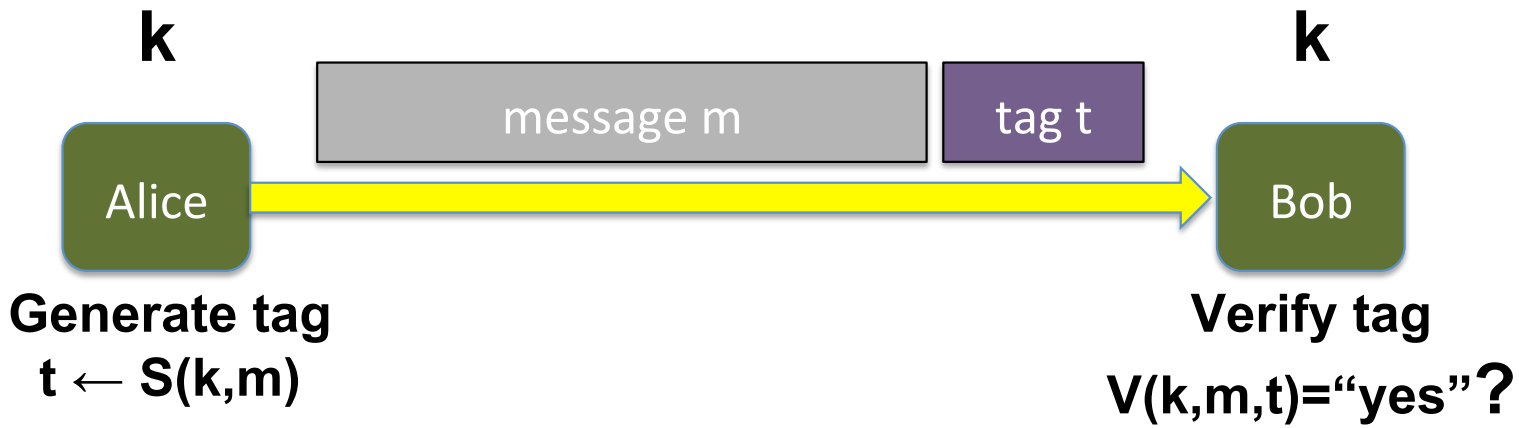
Goal

Data integrity and origin authenticity in the public-key setting



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

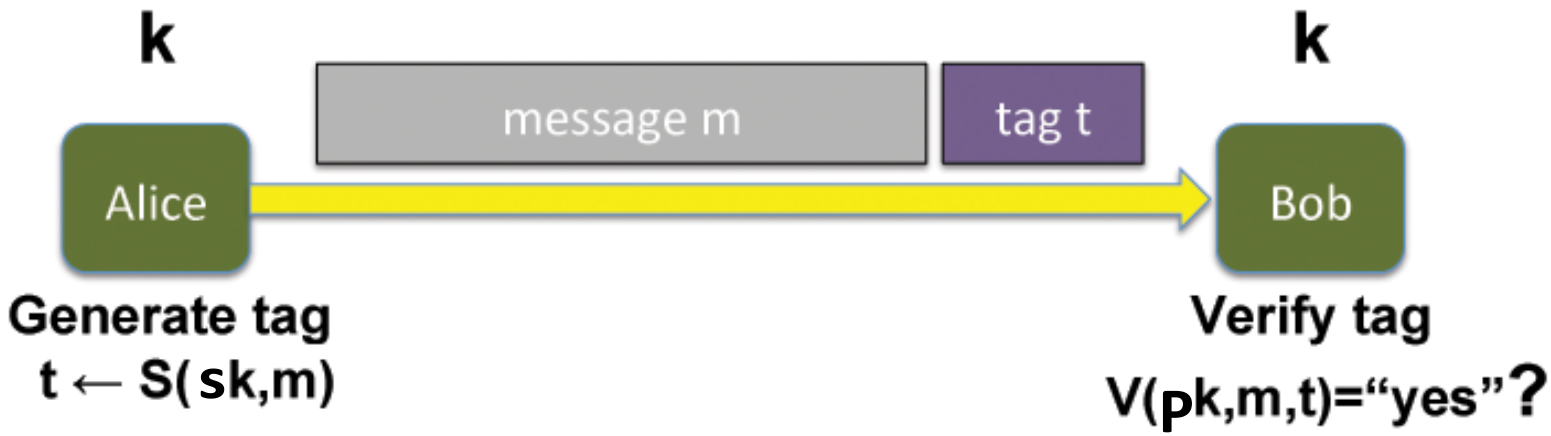
Advantages of digital signatures over MACs



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 her secret key, she cannot deny it later

Security

A good digital signature schemes should satisfy existential unforgeabitliy.

Existential unforgeability

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

Textbook RSA signatues

- $G_{RSA}() = (pk, sk)$ 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: $V_{RSA}(pk, m, x) = \begin{cases} \top & \text{if } m = x^e \pmod{N} \\ \bot & \text{otherwise} \end{cases}$

  where $sk = (N, d)$
- st $\forall (pk, sk) = G_{RSA}(), \forall x, V_{RSA}(pk, x, S_{RSA}(sk, x)) = \top$
  Proof: exactly as proof of consistency of RSA encryption/decryption

Problems with "textbook RSA sinatures"

Textbook RSA sinatures are not secure

The "textbook RSA sinature" scheme does not provide existential unforgeabitlity

- 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 signaure

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

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: $V_{RSA}(pk, m, x) = \begin{cases} \top & \text{if } H(m) = x^e \pmod N \\ \bot & \text{otherwise} \end{cases}$

Cryptography: public key infrastructure

Myrto Arapinis
School of Informatics
University of Edinburgh

February 4, 2019

Public keys

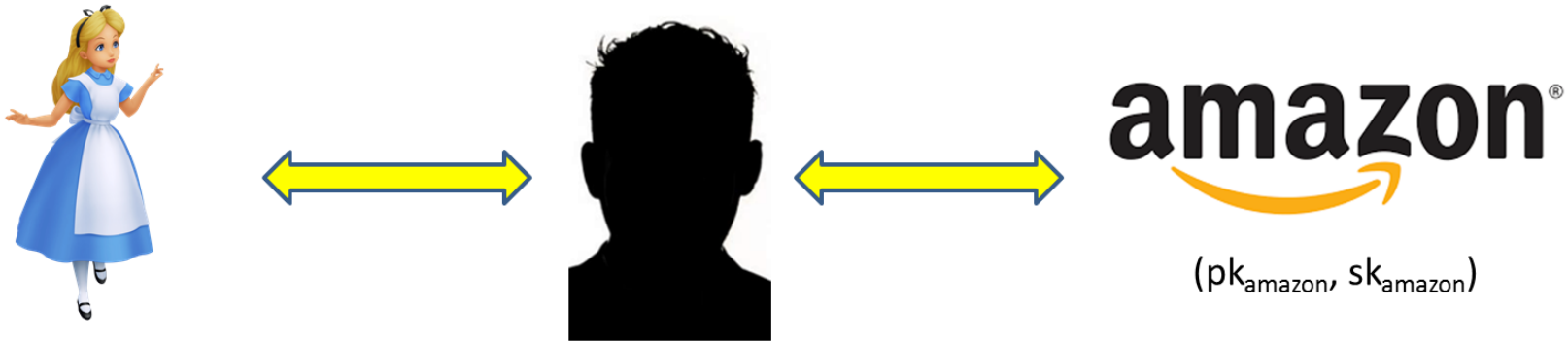


Figure: How does Alice trust that pk_{Amazon} is Amazon's public key?

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 forgeries

2. Publicly available directories - participants publish their public key on public directories
   ☹ does not defend against forgeries

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

Public key certificates

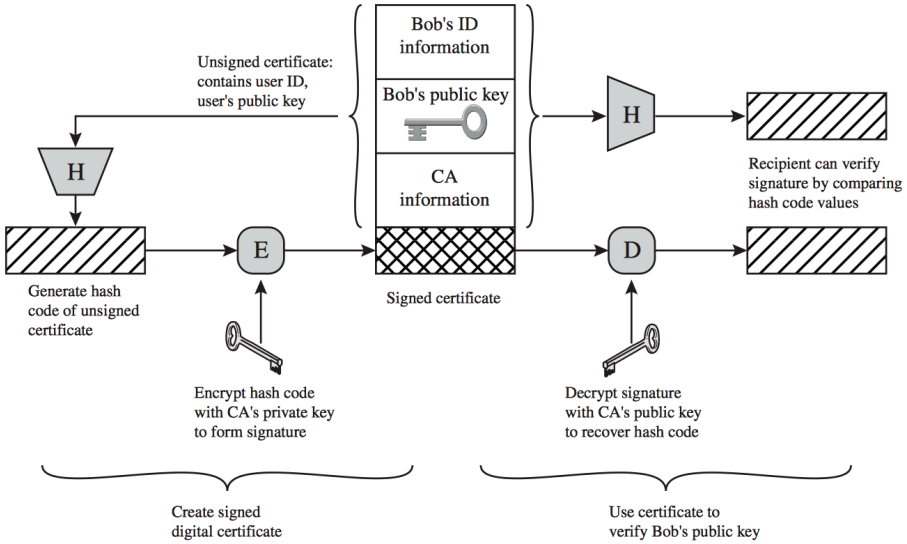


Figure: image from Cryptography and Network Security - Principles and Practice - William Stallings

A certificate consists mainly of

- a public key
- a subject identifying the owner of the key
- a signature by the CA on the key and the subject binding them together
  the CA is trusted

X.509 certificates

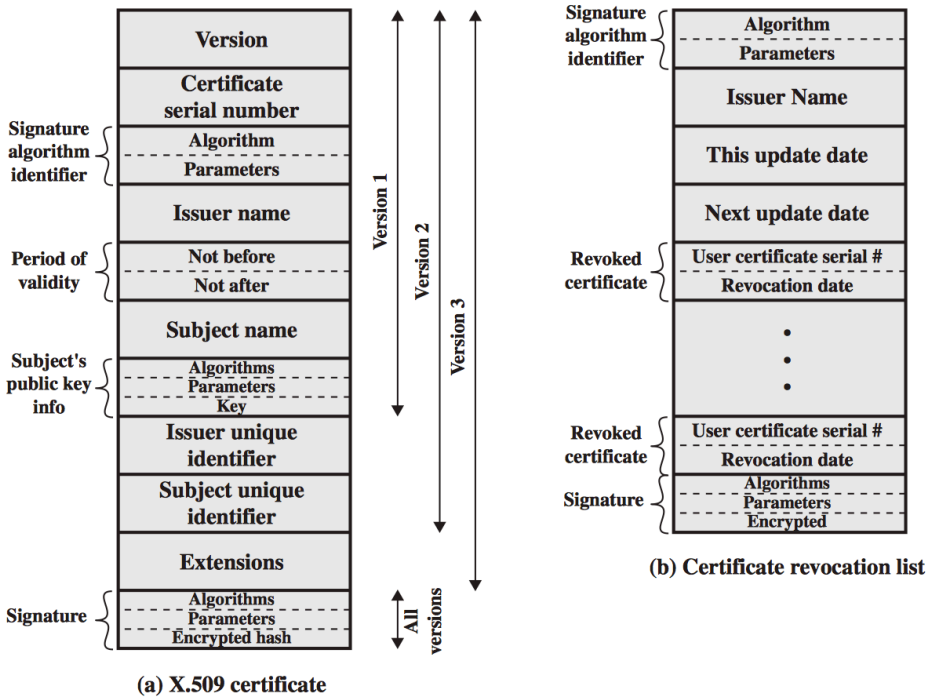


Figure: image from Cryptography and Network Security - Principles and Practice - William Stallings

- X.509 defines a framework for the provision of authentication services
- Used by many applications such as TLS

Public key certificates

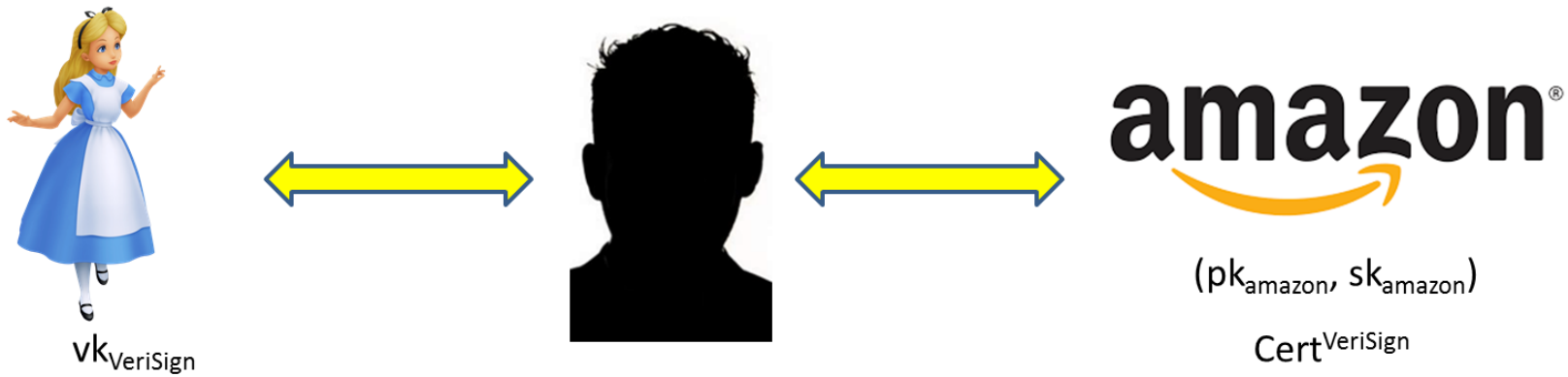
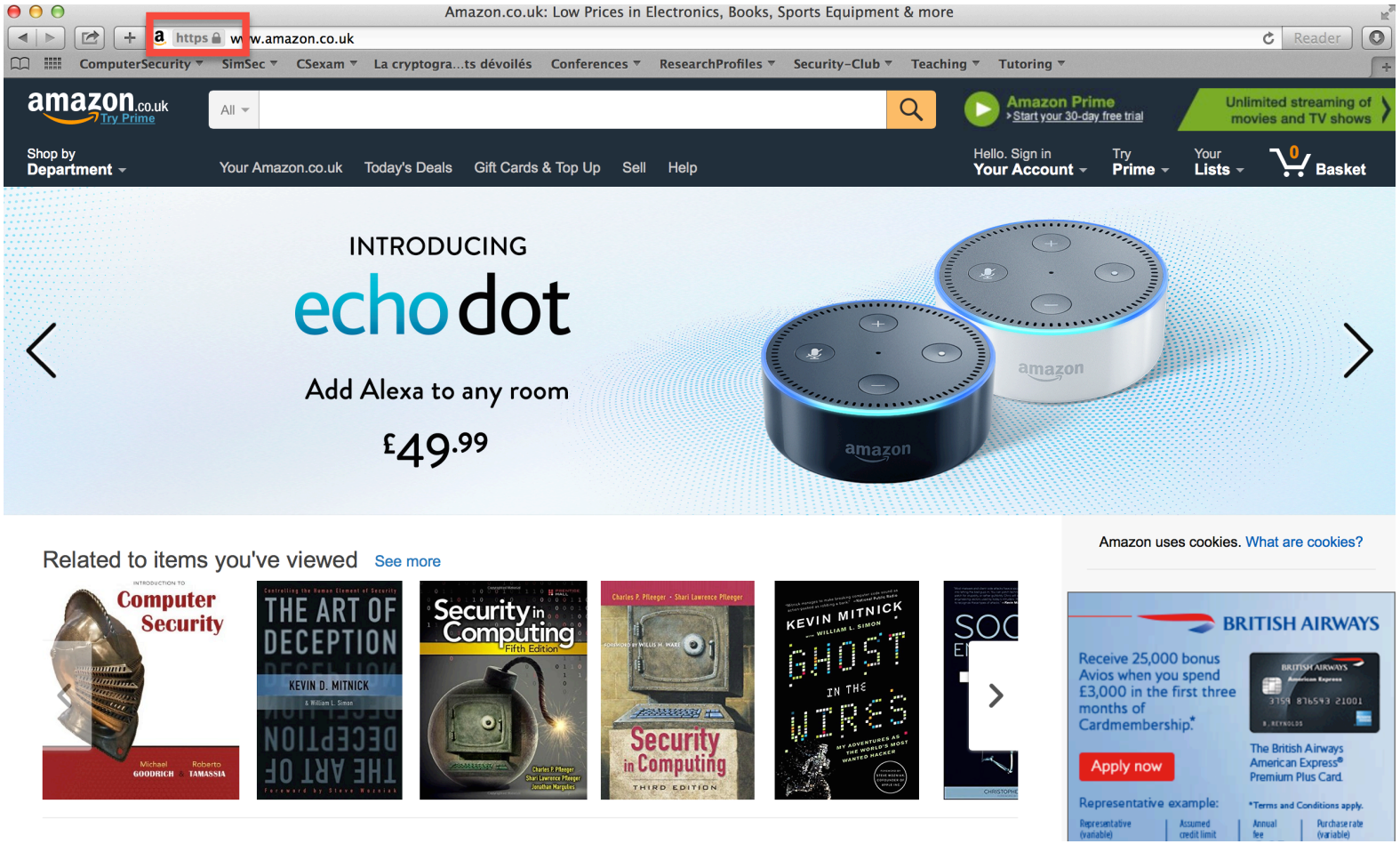


Figure: Alice can now verify Amazon's certificate

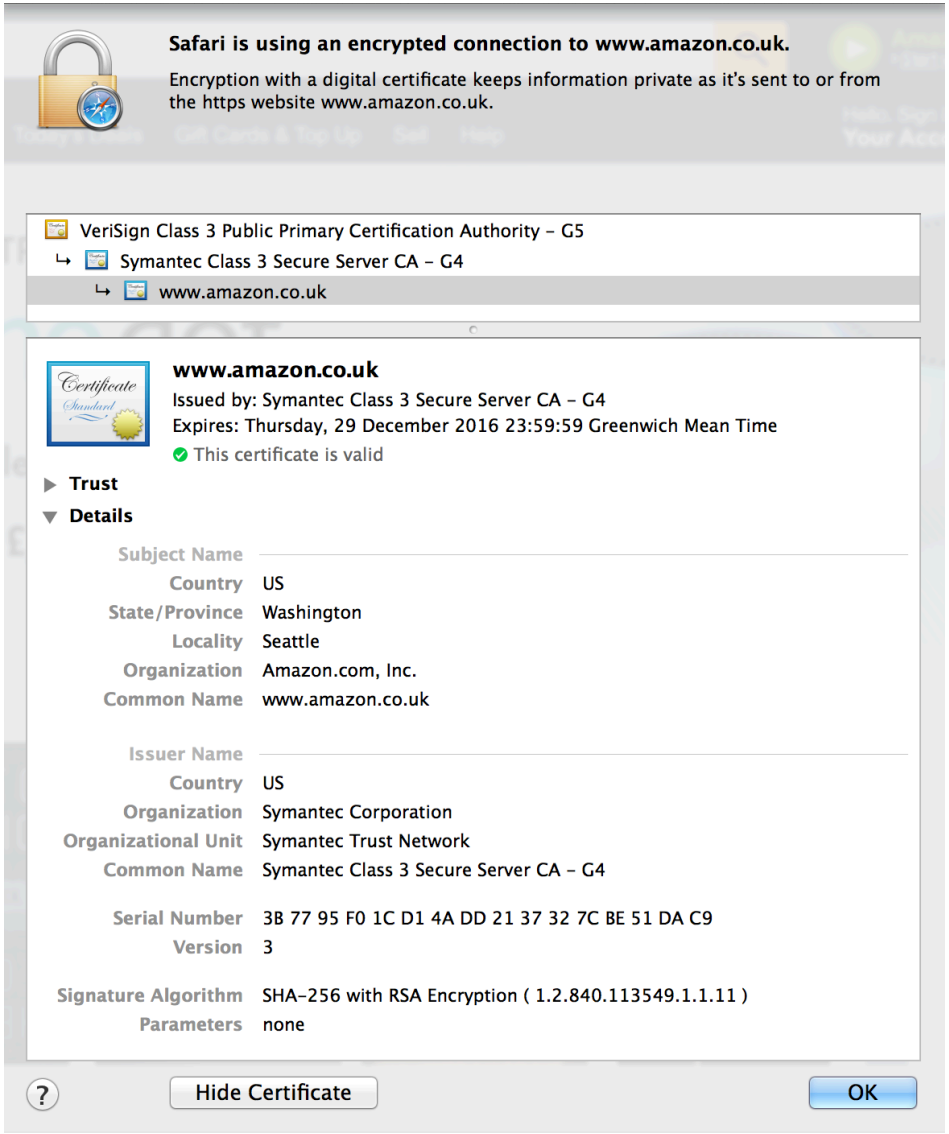
Using public key certificates to secure the Internet



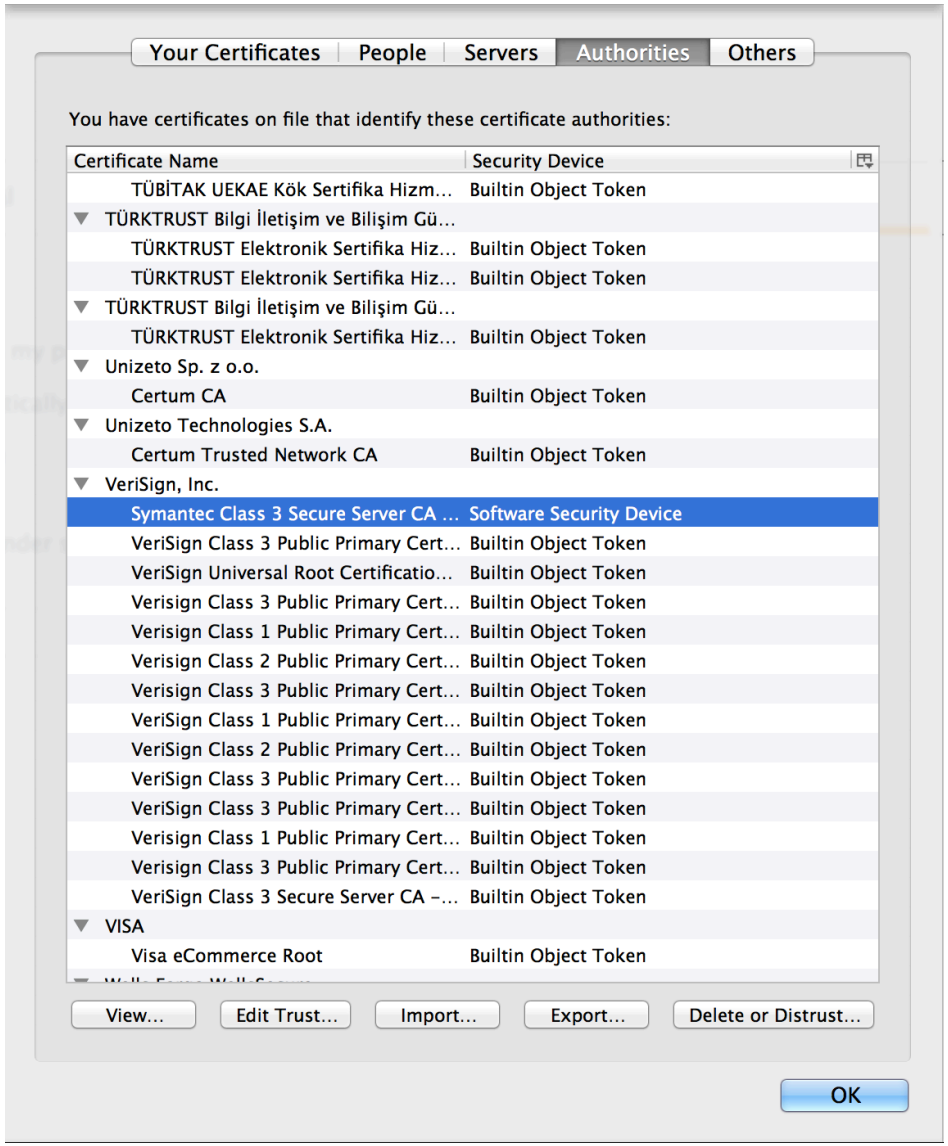
A very important implicit assumption

The browser is trusted to be "secure"

Amazon's certificate



Browser root certificates



Chain of trust

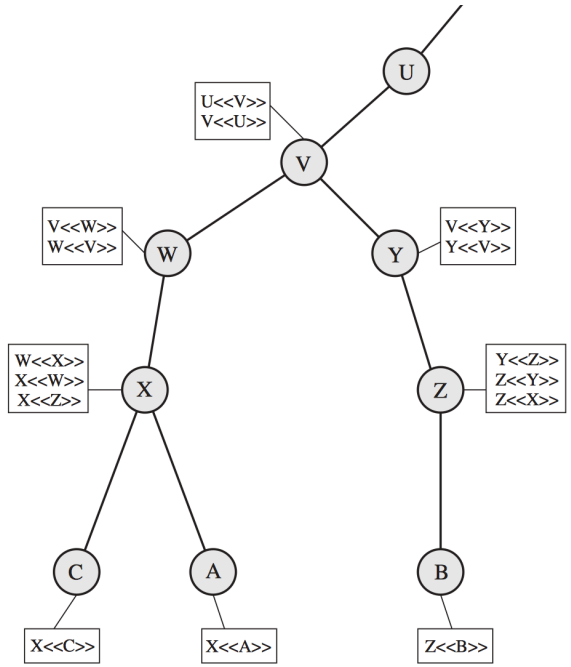
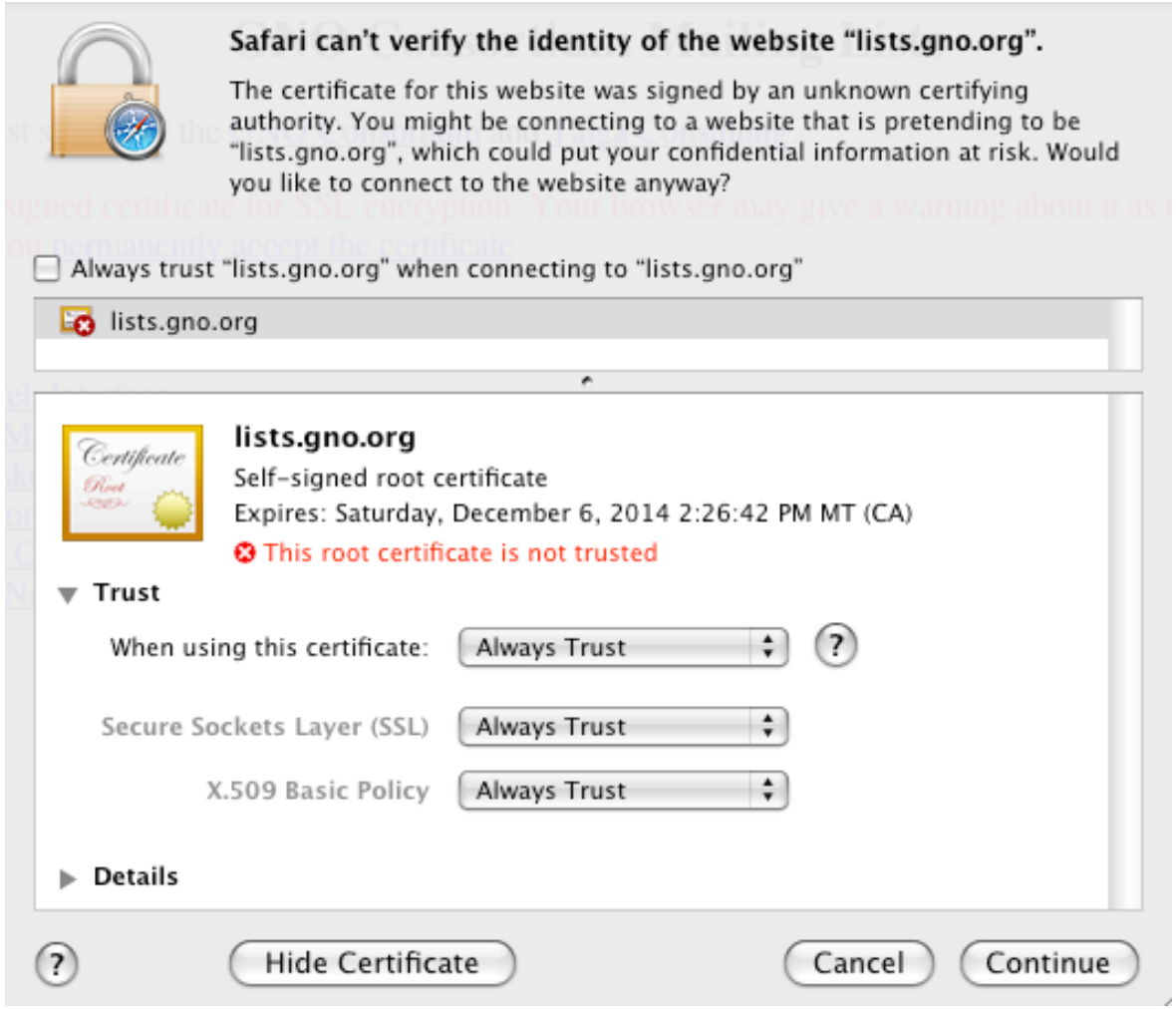


Figure: X.509 Hierarchy - image from Cryptography and Network Security - Principles and Practice - William Stallings

- Having a single CA sign all certificates is not practical
- Instead a root CA signs certificates for level 1 CAs, level 1 CAs sign certificates for level 2 CAs, *etc*

Self-signed certificates



The Lenovo Superfish scandal (February 2015)

c|net Search Reviews News Video How To Smart Home Cars Deals Dow

CNET › Security › Lenovo's Superfish security snafu blows up in its face

Lenovo's Superfish security snafu blows up in its face

The preloaded Superfish adware does more than hijack website ads in a browser. It also exposes Lenovo owners to a simple but dangerous hack that could spell disaster.

Security

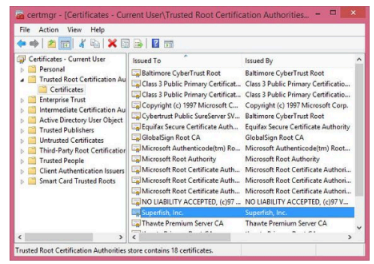
February 20, 2015
5:00 AM PST

by Seth Rosenblatt
@sethr

Removing software that comes with your brand-new Windows computer can be frustrating, but recently discovered software on new Lenovo laptops -- the top-selling laptop brand in 2014 -- can put your entire digital life at risk.

The preloaded software, called Superfish, alters your search results to show you different ads than you would otherwise see. But it also tampers with your computer's security so that attackers can snoop on your browser traffic -- no matter which browser you're using.

"Attackers are able to see all the communication



Superfish code hides in hard-to-reach places on your new Lenovo laptop, making it difficult to remove.

Screenshot by Robert Graham/Errata Security

And more recently (September 2016)

The Register®
Biting the hand that feeds IT

DATA CENTRE SOFTWARE NETWORKS SECURITY TRANSFORMATION DEVOPS BUSINESS HARDWARE SCIENCE

Security

Mozilla wants woeful WoSign certs off the list

Backdating SHA-1 certs is just not on

27 Sep 2016 at 03:58, Richard Chirgwin 42

Mozilla wants to kick Chinese certificate authority (CA) WoSign out of its trust program.

More

Mozil

IT
So
Tick
Mgr
Serv
freshs