PKI Applications: Web security

Université Grenoble Alpes

Jean-Guillaume Dumas

PKI applications: Web security

1. Respective situations of security systems in the protocols realm
2. SIGMA: Authenticated Diffie-Hellman
3. IPSec
4. SSL/TLS
   – OpenSSL
5. E-mail
   – S/MIME
   – PGP and GnuPG
   – DNSsec
   – OTR
6. Secure payments
   – SET: Secure Electronic Transactions
   – 3D-Secure
   – e-IDAS
   – Bitcoin
7. LDAP

Respective situations of security systems in the protocols realm

1. Network layer

smtp	ftp	http
TCP		
IP / IPSec		

2. Transport layer

smtp	ftp	http
SSL or TLS		
TCP		
IP		

3. Application layer

	S/mime	PGP	SET/EMV
Kerberos	smtp	http/https	
UDP	TCP		
IP			

PKI applications: Web security

1. Respective situations of security systems in the protocols realm
2. SIGMA: Authenticated Diffie-Hellman
3. IPSec
4. SSL/TLS
   – OpenSSL
5. E-mail
   – S/MIME
   – PGP and GnuPG
   – DNSsec
   – OTR
6. Secure payments
   – SET: Secure Electronic Transactions
   – 3D-Secure
   – e-IDAS
   – Bitcoin
7. LDAP

2. Authenticated Diffie-Hellman : SIGMA

A. Basic Authenticated D-H (BAD-H)
   – Man-in-the-middle protection

B. ISO-9706 protection against identity misbinding

C. Station-To-Station: proof of knowledge of the key

D. SIGMA: Sign and MAC
   ✚ identity misbinding protection
   ✚ Key binding to the peer

E. Examples

Basic Authenticated D-H (BAD-H)

<u>Alice</u>		<u>Bob</u>
1) $A \,\|\|\, g^x$	-------->	
	<--------	2) $B \,\|\|\, g^y \,\|\|\, SIG_B(g^x,g^y)$
3) $SIG_A(g^x,g^y)$	-------->	

Identity Misbinding *[Unkown Key-Share attack]*

<u>Alice</u>		<u>Eve</u>		<u>Bob</u>
$A\|\|g^x$	-------->	$E\|\|g^x$	-------->	
	<--			$B\|\|g^y\|\|SIG_B(g^x,g^y)$
$SIG_A(g^x,g^y)$	-------->	$SIG_E(g^x,g^y)$	-------->	

⇨ E doesn't know K, but B considers anything sent by A as coming from E

ISO-9706 protection

<u>Alice</u>		<u>Bob</u>
1) $A \,\|\|\, g^x$	-------->	
	<--------	2) $B \,\|\|\, g^y \,\|\|\, SIG_B(g^x,g^y,A)$
3) $SIG_A(g^x,g^y,B)$	-------->	

Protection against active attacker is not guaranteed

- ☝ With E in the middle, B would send **$SIG_B(g^x,g^y,E)$**
  - ✓ E cannot produce **$SIG_B(g^x,g^y,A)$**
  - ⇨ **A stops the protocol and does no use the key**
- ☺ The ISO protocol avoids the misbinding attack

- △ Protection against active attacker is not possible
- ☹ B needs to know the identity of A beforehand

Station-To-Station protection

<u>Alice</u>		<u>Bob</u>
1) g^x	-------->	
	<--------	2) $g^y \,\|\|\, E_K(B, SIG_B(g^x,g^y))$
3) $E_K(A, SIG_A(g^x,g^y))$	-------->	

- ☝ Proof of knowledge of K
- ☺ Peer's ID is not needed for own authentication
- ☺ Can covers identities or certificates

- △ Encryption is not really the right function to prove knowledge of a key
- △ Original STS (even STS-MAC) did not include the identity
  - ⇨ Back to identity misbinding

SIGMA (sign and MAC) protocol

Alice		Bob
g^x	-------->	
	<--------	$B \| g^y \| SIG_B(g^x,g^y) \| MAC_K(B)$
$A \| SIG_A(g^x,g^y) \| MAC_K(A)$	-------->	

☺ Signature: man-in-the-middle protection
☺ Identity: identity binding protection
☺ MAC: proof of knowledge + binding of the key to the identity

⇨ SIGN and MAC your OWN identity

? ID protection against active attacks ?

SIGMA-I active protection of Initiator's ID

Alice		Bob
g^x	-------->	
	<--------	$g^y \| E_{Ke}(B \| SIG_B(g^x,g^y) \| MAC_{Km}(B))$
$E_{Ke}(A \| SIG_A(g^x,g^y) \| MAC_{Km}(A))$	-------->	

⇨ Ke and Km are derived from $K = g^{xy}$

☺ Alice does not reveal her identity until she is sure that only Bob can retreive it

- △ An active attacker could replay message 2
- ☺ But cannot retrieve the initiator's ID from message 3 since he cannot complete the Diffie-Hellman computation (new x, old y)

SIGMA-R active protection of Responder's ID

Alice		Bob
g^x	-------->	
	<--------	g^y
$E_{Ke}(A \| SIG_A(g^x,g^y) \| MAC_{Km}(A))$	-------->	
	<--------	$E_{ke'}(B \| SIG_B(g^x,g^y) \| MAC_{Km'}(B))$

☺ Bob does not reveal his identity until he is sure that only Alice can retrieve it

☺ $(Ke,Km) \neq (Ke',Km')$: against reflection attack

SIG-MAC Exemples

- OID in certificates: DHE_RSA; ECDHE_RSA; ECDHE_ECDSA …
- SIGMA-R
  - $E_{Ke}(A||SIG_A(g^x,g^y)||MAC_{Km}(A))$
- SSL hand-shake
  - CertificateVerify : digitally-signed SHA(master_secret + h-s msg …);
  - Finished : SHA(master_secret + h-s msg …);
- IPsec IKE
  - $B \,||\, SIG_B(\, MAC_{Km}(B,g^x,g^y)\,)$ in aggressive mode (no ID protection)
  - $E_{Ke}(A||SIG_A(MAC_{Km}(A,g^x,g^y))\,)$ in main mode (responder ID protection)
- OTR AKE
  - $E_{Ke}(A||SIG_A(MAC_{Km1}(A,g^x,g^y)\,) \,||\, MAC_{Km2}(E_{Ke}(\dots)\,)$

△ WARNING
- △ B || SIG || MAC (SIG) is still subject to key misbinding attack (STS-MAC)
- ⇨ must add ID inside MAC to bind proof of K to identity

PKI applications: Web security

1. Respective situations of security systems in the protocols realm
2. SIGMA: Authenticated Diffie-Hellman
3. IPSec
4. SSL/TLS
   – OpenSSL
5. E-mail
   – S/MIME
   – PGP and GnuPG
   – DNSsec
   – OTR
6. Secure payments
   – SET: Secure Electronic Transactions
   – 3D-Secure
   – e-IDAS
   – Bitcoin
7. LDAP

IPSec [RFC 2410, 4301-3]

- *IKEv2 (Internet Key Exchange) [RFC 4306]*: is the PKI managing the setting-up of contacts, recognition of contacts and interchange of IPSec security keys

  It makes use of known algorithms:
  – Encryption: DES and 3DES ; AES
  – Public keys: DSA, RSA, Diffie-Hellman
  – Hashing (imprint): MD5 and SHA-?

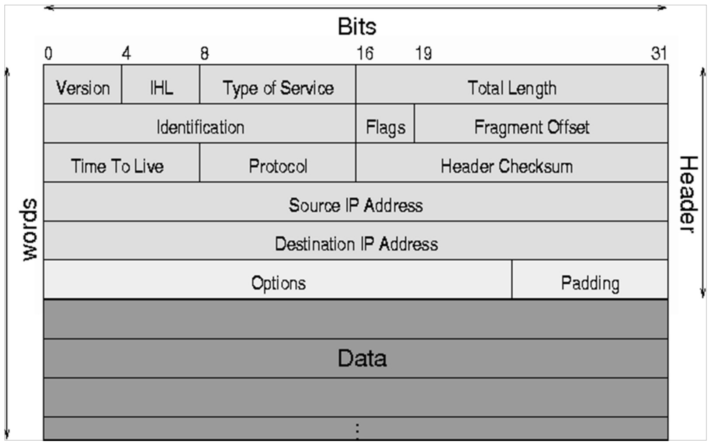
  http://www.ietf.org/internet-drafts/draft-ietf-pki4ipsec-ikecert-profile-11.txt

- During interchanges, IPSec offers two levels of security, by virtue of two sub-protocols:
  – *AH (Authentication Header)*: authentication header that guarantees the integrity and origin of the data, but does not provide confidentiality.
  – *ESP (Encapsulating Security Payload)*: apart from AH functions, ensures confidentiality via encapsulation.

- In practice, messages using PKIs can travel in IPSec tunnels.

IPSec packets

- IP packet



- IPSec tunnel

New header	IPSec header	Original IP header	IP data

- IPSec transport

Original IP header	IPSec header	IP data

IPSec security

- AH (Authentification Header)
  – IPSec tunnel

New header	AH header	Original IP header	IP data

  – IPSec transport

Original IP header	AH header	IP data

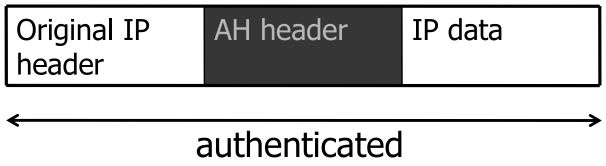
- ESP (Encapsulating Security Payload)

Original IP header	ESP header	IP data	ESP trailer	Authentication data

Original IP header	ESP header	IP data	ESP trailer

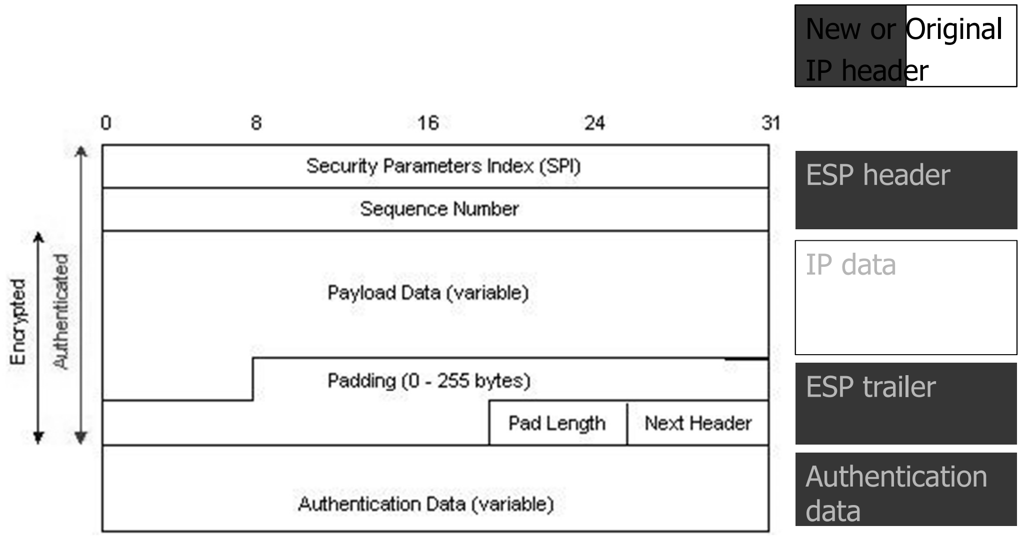
etc.

AH controversy

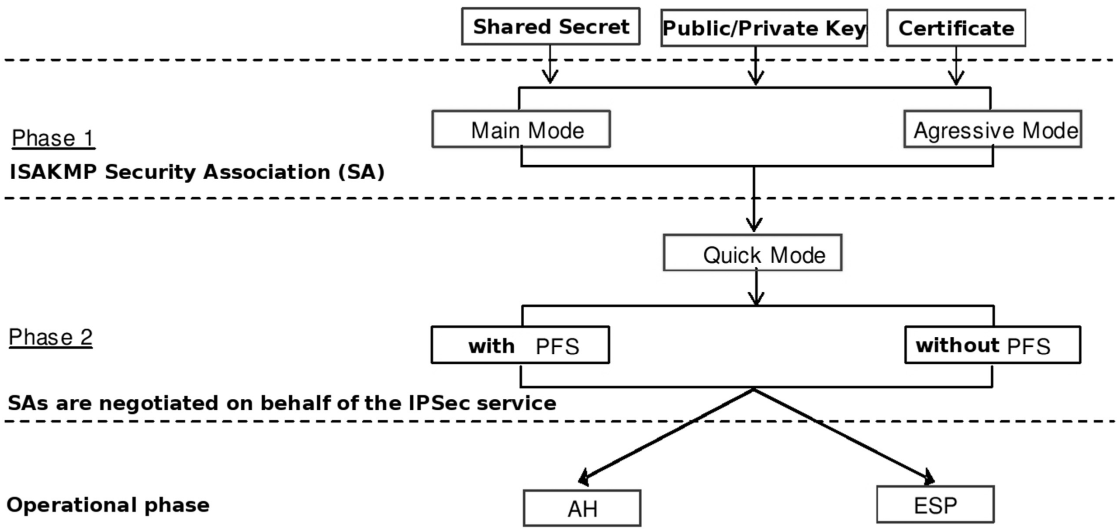


- Authentication provided by ESP as well
  - but not on the original IP header …
  - … protecting immutable fields does not add much
- AH not efficient to compute (MAC at the beginning)

Encapsulating Security Payload



Internet Key Exchange (IKE) the PKI of IPsec



PFS (perfect forward secrecy): compromission of long-lived keys does not compromise any previous conversation

ISAKMP Security Association (IKEv1)

Initiator		Responder
1) $Hdr \| \| SA_{I1}$	----------------->	
	<-----------------	2) $Hdr \| \| SA_{R1}$
3) $Ke_I \| \| N_I$	----------------->	
	<-----------------	4) $Ke_R \| \| N_R$
D-H: skeyseed		*D-H: skeyseed*
5) $ID_1 \| \| [Cert_I] \| \| Sign_I$	----------------->	
	<-----------------	6) $ID_R \| \| [CERT_r] \| \| Sign_R$

- Hdr: security parameters, version numbers, etc.
- SA_I, SA_R: supported algorithms
- Ke_I, Ke_R: Diffie-Hellman values
- N_I, N_R: nonces
- $Cert_I$, $Cert_R$: optional certificates
- $Sign_I = E_{KprivI}(h(ID_I || N_R || Ke_I))$; $Sign_R = E_{KprivR}(h(ID_R || N_I || Ke_R))$

PKI applications: Web security

1. Respective situations of security systems in the protocols realm
2. SIGMA: Authenticated Diffie-Hellman
3. IPSec
4. SSL/TLS
   – OpenSSL
5. E-mail
   – S/MIME
   – PGP and GnuPG
   – DNSsec
   – OTR
6. SET: Secure Electronic Transactions
7. LDAP

SSL / TLS [RFC5246]

- Connection: status
  – Client challenge + Server challenge
  – $K_{privServer}$ + $K_{privClient}$ for signatures
  – K_{Server} Server symmetrical encryption
  – K_{Client} Client symmetrical encryption
  – IV for CBC mode
  – Message numbers

Hand-shake	Change cipher spec	Alert	http
SSL record protocol			

- Connection: transport
- Session == (client <--> server association)
  – Session ID
  – Certificate of the X509 v3 contact
  – Compression method
  – Encryption specification
  – 48-byte master secret
  – Re-use (0 or 1) → new connections?

SSL handshake: Authentification + ciphering algorithm/MAC + keys

<u>Client</u>		<u>Server</u>
1) ClientHello	-------->	2) ServerHello
		3) Certificate*
		4) ServerKeyExchange*
		5) CertificateRequest*
	<--------	6) ServerHelloDone
7) Certificate*		
8) ClientKeyExchange		
9) CertificateVerify*		
10) [ChangeCipherSpec]		
11) Finished	-------->	
		12) [ChangeCipherSpec]
	<--------	13) Finished
Application Data	<------->	Application Data

SSL Handshake

- struct {
  – HandshakeType msg_type; /* type of handshake message */
  – uint24 length; /* # bytes in handshake message body */
  – select (HandshakeType) {
    - case hello_request: HelloRequest;
    - case client_hello: ClientHello;
    - case server_hello: ServerHello;
    - case certificate: Certificate;
    - case server_key_exchange: ServerKeyExchange;
    - case certificate_request: CertificateRequest;
    - case server_hello_done: ServerHelloDone;
    - case certificate_verify: CertificateVerify;
    - case client_key_exchange: ClientKeyExchange;
    - case finished: Finished;

    } body;

  } Handshake;
- struct { uint32 gmt_unix_time; opaque random_bytes[28]; } Random;
- struct {
  – opaque dh_p<1..2^{16}-1>;
  – opaque dh_g<1..2^{16}-1>;
  – opaque dh_Y_s<1..2^{16}-1>;

  } ServerDHParams; /* Ephemeral DH parameters */
- struct {
  - opaque rsa_modulus<1..216-1>;
  - opaque rsa_exponent<1..216-1>;

  } ServerRSAParams;

Some SSL handshake messages

- struct { ProtocolVersion client_version; Random random;
  SessionID session_id; CipherSuite cipher_suites<2..2^{16}-1>;
  CompressionMethod compression_methods<1..2^8-1>;
  } xxxxxxHello;

…

- struct { ASN.1Cert certificate_list<1..2^{24}-1>; } Certificate;

…

- struct { select (KeyExchangeAlgorithm) {
  case rsa: EncryptedPreMasterSecret;
  case diffie_hellman: ClientDiffieHellmanPublic;
  case fortezza_dms: FortezzaKeys; } exchange_keys;
  } ClientKeyExchange;

- struct { ServerXXXParams params;
  digitally-signed hash[20] signed_params;
  } ServerKeyExchange;

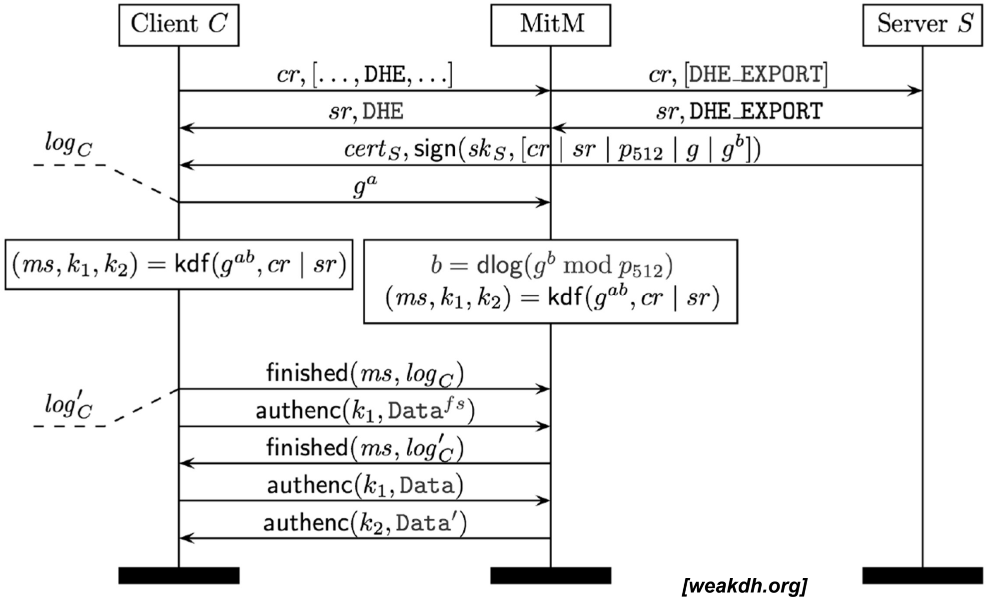
…

- struct {digitally-signed opaque hash[20]; } CertificateVerify;
  CertificateVerify.signature.sha_hash SHA (master_secret + pad2 +
  SHA(handshake_messages + master_secret + pad1)
  );

…

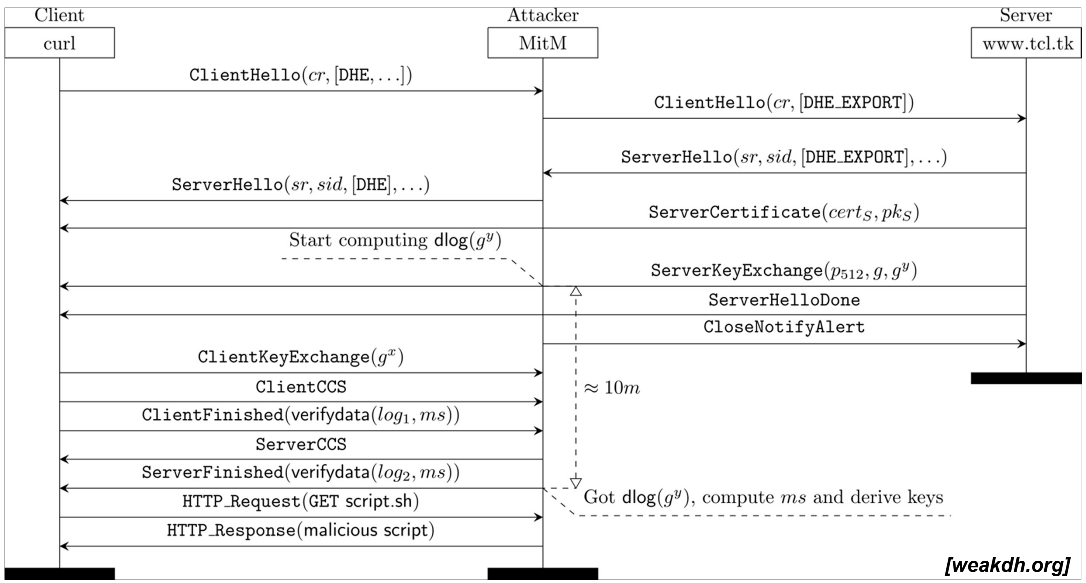
- struct { opaque sha_hash[20]; } Finished;
  Finished .sha_hash SHA (master_secret + pad2 +
  SHA(handshake_messages + Sender + master_secret + pad1)
  );

Logjam attack: Man-in-the-middle with weak DH key (DHE_EXPORT)



[weakdh.org]

Logjam attack: 512 bits D-H group !

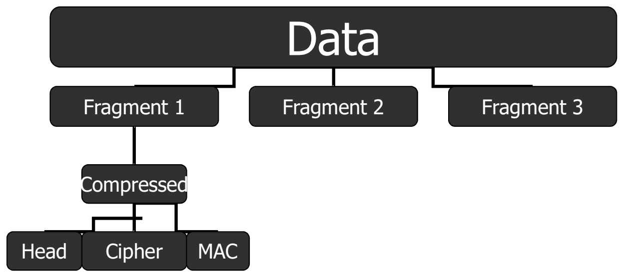


[weakdh.org]

SSL alerts and data

- Alerts: two compressed and encrypted bytes
  - Warning (close-notify; no-certificate; bad-certificate; certificate-revoked; certificate-expired; etc.)
  - Fatal (unexpected-message; bad-record-mac; decompression-failure; handshake-failure; illegal parameter; etc.)

- Record protocol
  1. Fragmentation
  2. Compression
  3. Addition of a MAC
  4. Ciphering
  5. SSL header



OpenSSL

- `openssl genrsa`: create keys
- `openssl enc`: encrypt
- `openssl rsautl`: sign
- `openssl dgst`: hashing
- `openssl req`: generate certification requests and self-signed certificates
- `openssl verify ; openssl x509 ; openssl crl`
- `openssl ca`: certify
- `openssl pkcs12`: conversions
- `openssl smime`

PKI applications: Web security

1. Respective situations of security systems in the protocols realm
2. SIGMA: Authenticated Diffie-Hellman
3. IPSec
4. SSL/TLS
   - OpenSSL
5. E-mail
   - S/MIME
   - PGP and GnuPG
   - DNSsec
   - OTR
6. Secure payments
   - SET: Secure Electronic Transactions
   - 3D-Secure
   - e-IDAS
   - Bitcoin
7. LDAP

S/MIME: MIME + PKI

- SMTP [RFC 822] (Simple Mail Transfer Protocol)
  - Format of messages sent by e-mail
    - From: … To: … Date: … etc.
- MIME [RFC 2025] (Multipurpose Internet Mail Extension)
  - Content-type; Content-Transer-Encoding; etc.
- S/MIME [RFC 2119]
  - pkcs7-mime: encapsulated data
  - pkcs7-mime: signed data
  - pkcs7-mime: certificates
  - pkcs7-signature: signature sub-part
  - pkcs10-mime: registration certificate request
  - pkcs12: personal information exchange syntax standard

Cryptographic Message Syntax [RFC2630]

- pkcs7-mime
  - EnvelopedData packet for ciphered messages
  - SignedData packet signed messages
- EnvelopedData
  - Several RecipientInfo structures
    - Receiver ID
    - $E_{pub\text{-}receiver}(K_{session})$
    - Used Algorithms
  - "inner" message: $E_{Ksession}$ (TextMessage)
  - △ In most mailers
    - One RecipientInfo for the sender too !!!
    - ⇨ This allows transparent on-the-fly decryption of the stored sent message
- SignedData
  - Certificates
  - Signature value

PKI applications: Web security

1. Respective situations of security systems in the protocols realm
2. SIGMA: Authenticated Diffie-Hellman
3. IPSec
4. SSL/TLS
   - OpenSSL
5. E-mail
   - S/MIME
   - PGP and GnuPG
   - DNSsec
   - OTR
6. Secure payments
   - SET: Secure Electronic Transactions
   - 3D-Secure
   - e-IDAS
   - Bitcoin
7. LDAP

PGP and electronic mail

- Confidentiality + authentication
- E-mail compatibility
  - Mail system: ASCII blocks
  - Radix-64: 8-bit binary conversion into printable ASCII stream
  - Segmentation
    - PGP overcomes the size limitation of increasing small e-mails "after all"
    - On reception, PGP removes the mail headers and reassembles the original block "before anything else"
- Problem: public key identifier?
  - 1 message: two public key identifiers – recipient and sender
  - Keyrings
    - Secret: K = Hash(Passphrase) destroyed and rebuilt during each usage
    - Public: storage of encrypted private keys

GnuPG

- Open source implementation of OpenPGP
- Special features:
  - Signatures of a public key with four trust levels
    0 I don't know
    1 I don't trust it
    2 I trust it somewhat
    3 I trust it totally
  - Local keyring
  - Revocation certificate
  - Signature + encryption of emails
  - Web of Trust (pkds key propagation protocol)
    http://pgp.mit.edu ; http://www.stinkfoot.org:11371/ ; http://pgp.zdv.uni-mainz.de/pks-commands.html ; http://keyserver.linux.it/ ; http://keyserver.stack.nl/ ; http://keyserver.oeg.com.au/ ; http://keys.niif.hu/ ; http://www.rediris.es/cert/servicios/keyserver/ ; http://kerckhoffs.surfnet.nl/ ;

PGP signature trustlevels

0 means you make no particular claim as to how carefully you verified the key.

1 means you believe the key is owned by the person who claims to own it but you could not, or did not verify the key at all. This is useful for a "persona" verification, where you sign the key of a pseudonymous user.

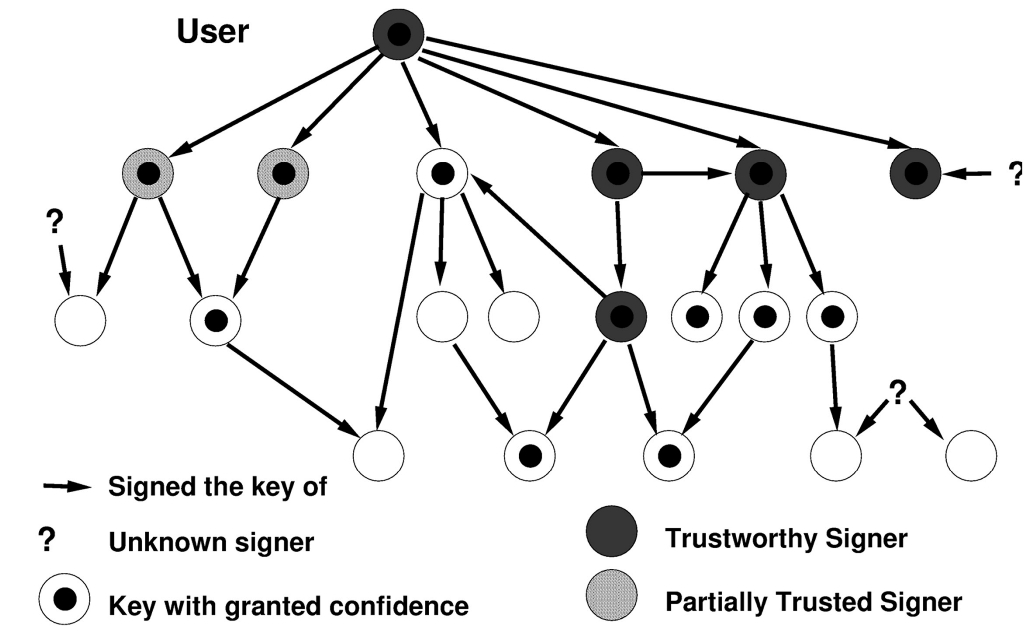
2 means you did casual verification of the key. For example, this could mean that you verified the key fingerprint and checked the user ID on the key against a photo ID.

3 means you did extensive verification of the key. For example, this could mean that you verified the key fingerprint with the owner of the key in person, and that you checked, by means of a hard to forge document with a photo ID (such as a passport) that the name of the key owner matches the name in the user ID on the key, and finally that you verified (by exchange of email) that the email address on the key belongs to the key owner.

GPG list of commands

- gpg --gen-key
- gpg --export [info-clef]
- gpg --import [fichier]
- gpg --list-keys
- gpg --list-sigs
- gpg --list-secret-keys
- gpg --delete-key [info-clef]
- gpg --delete-secret-key
- gpg --fingerprint
- gpg --edit-key [info-clef]
- gpg --gen-revoke [info-clef]
- gpg --encrypt destinataire [message]
- gpg [--decrypt] [message]
- gpg --sign [message]
- gpg --verify [message]

PGP: trust model



Private keyring

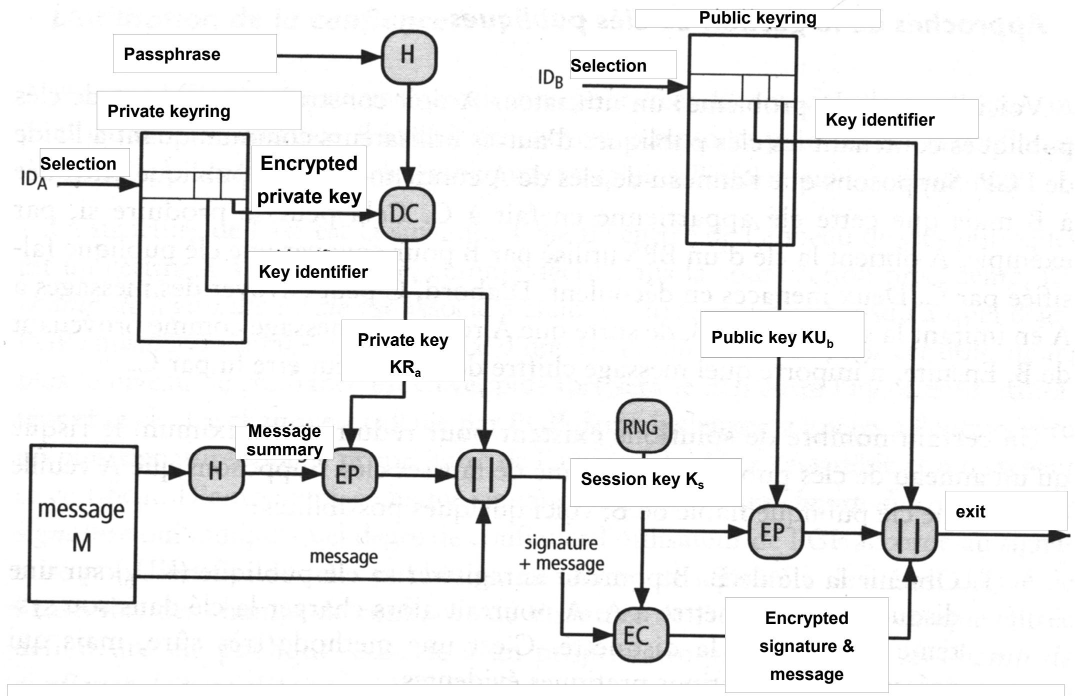
Time- and date-stamp	Key identifier	Public key	Encrypted private key	User identifier
⋮	⋮	⋮	⋮	⋮
T_i	$KU_i \bmod 2^{64}$	KU_i	$E_{H(P_i)}[KR_i]$	User i
⋮	⋮	⋮	⋮	⋮

Public keyring

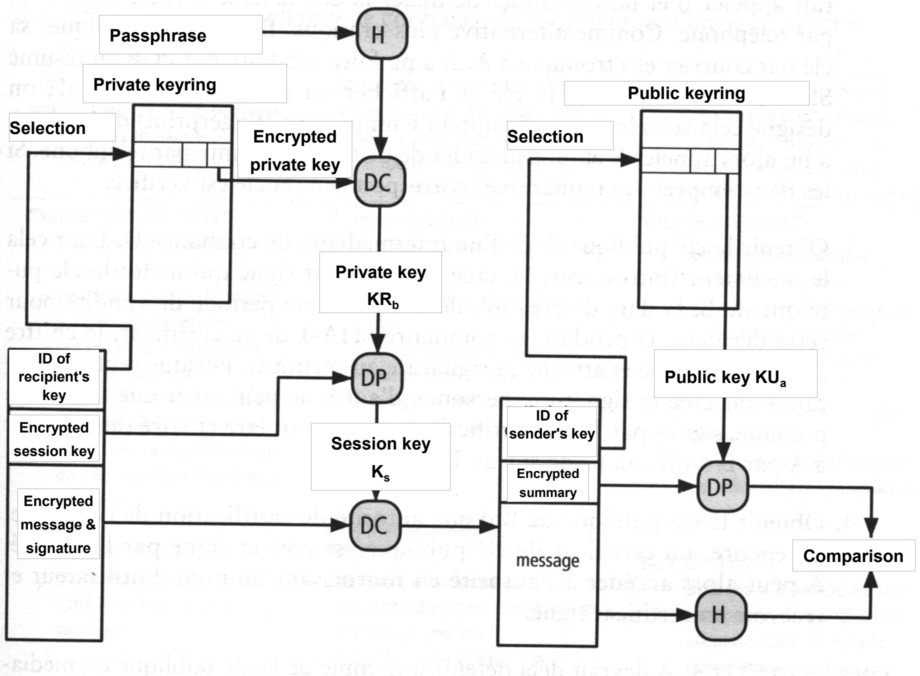
Time- and date-stamp	Key identifier	Public key	Trusted owner	User identifier	Key legitimacy	Signature(s)	Trusted signature(s)
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
T_i	$KU_i \bmod 2^{64}$	KU_i	trust_flag$_i$	User i	trust_flag$_i$		
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

* = Field used to index the table

General structure of public and private keyrings



Generation of PGP message (from user A to user B);no Radix-64 conversion or compression)



Reception of a PGP message (from user A to user B; no Radix-64 conversion or compression)

PGP versus X509

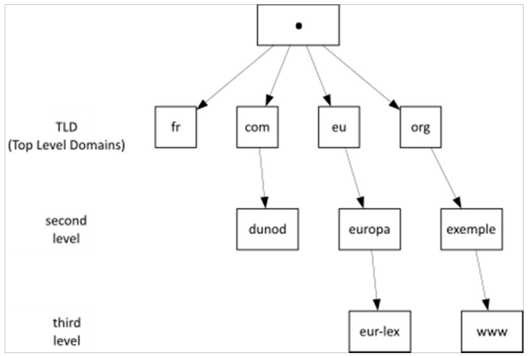
	PGP	X509
Certification authority	All users	1 only
Digital signature	Several	1 only
Key holder	Several	1 only
Revocation	Sender + those added by the sender as revocation authority	CA only

PKI applications: Web security

1. Respective situations of security systems in the protocols realm
2. SIGMA: Authenticated Diffie-Hellman
3. IPSec
4. SSL/TLS
   – OpenSSL
5. E-mail
   – S/MIME
   – PGP and GnuPG
   – DNSsec
   – OTR
6. Secure payments
   – SET: Secure Electronic Transactions
   – 3D-Secure
   – e-IDAS
   – Bitcoin
7. LDAP

DNS

- **DNS threats**
  – Packet interception
  – Cache corruption
  – ID prediction



DNSsec [RFC 4033-5]

- **DNSSEC** fights *DNS cache poisoning*
  - Authentication of the origin of DNS data
  - Data integrity
  - Authenticated existence denial
    - ⇨DNS Public Key (DNSKEY)
    - ⇨Resource Record Signature (RRSIG: validity, signed IP records)
    - ⇨Delegation Signer (DS: Signature by a CA)
    - ⇨Next Secure (NSEC: CRL)

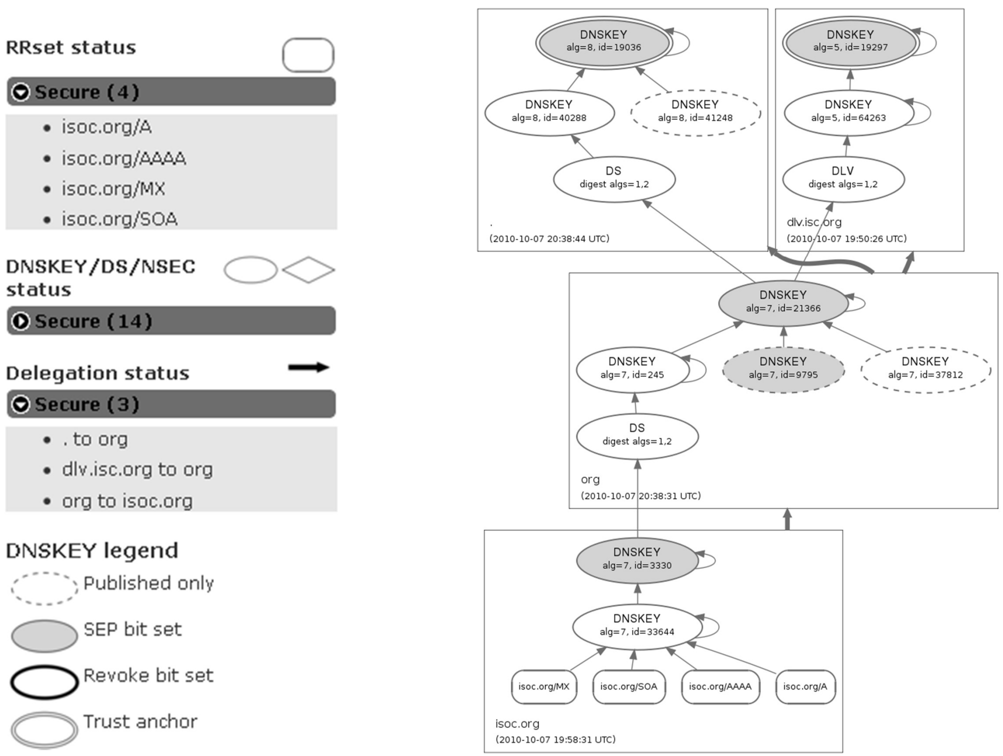
⇨ Checking Disabled (CD) and Authenticated Data (AD).

DNSSEC chain of trust

- Records digitally signed with public keys
  - Delegation Signer (DS) Record in a parent domain (DNS zone)
  - ⇨ Verifies DNSKEY record in a subdomain
    - ⇨DNSKEY record authenticated via chain of trust
    - ⇨Can contain other DS records to verify further subdomains

- Set of verified public keys
  - Within DNS root zone == trusted third party

1) DNS root (DNSKEY and DS for .com)
2) .com (DNSKEY and DS for example.com)
3) example.com (DNSKEY and RRSIG for www.example.com)

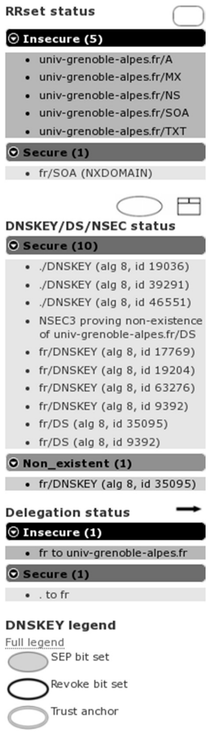


NSEC 3 [RFC 5155] for DNSSec

- NSEC: robust resistance against spoofing
  - △ Suppose there is no domain www.example.com
  - ☝ NSEC record instead of the RRSIG

- NSEC records allow the resolver to prove that a domain name does not exist
  - 😐 RRSIG records of ALL the names in a zone
  - ⇨ Proof of non-existence if a given name is not in the list
  - ☹ Problem: hostile party can enumerate all the names in a zone

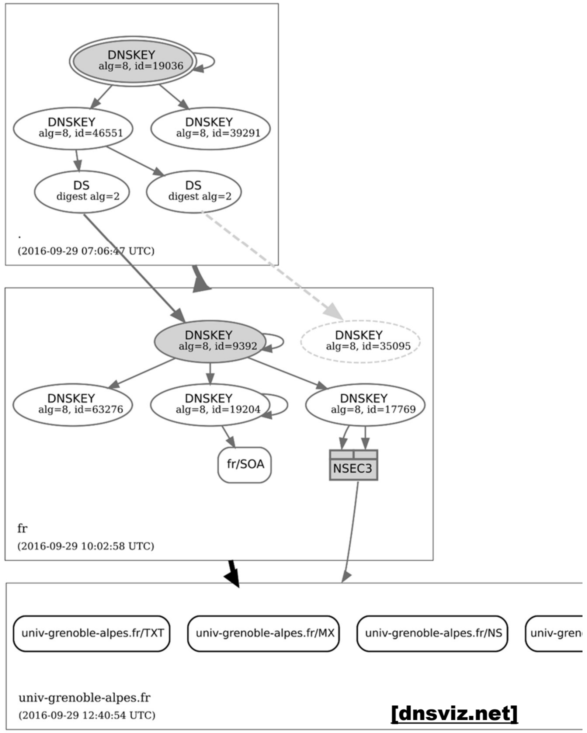
- NSEC3: Hashed zone enumeration
  - ☺ Hashed only and signed RR records of ALL the names
  - ⇨ Proof: hash www.example.com record, check if hash is in the list



DNSSec for univ-grenoble-alpes.fr

First DNS root zone: July 15th 2010

.com signed with valid DNSKEY and DS: April 1st 2011



[dnsviz.net]

DNSSec analyzer

Analyzing DNSSEC problems for univ-grenoble-alpes.fr

.	Found 3 DNSKEY records for . DS=19036/SHA-1 verifies DNSKEY=19036/SEP Found 1 RRSIGs over DNSKEY RRset RRSIG=19036 and DNSKEY=19036/SEP verifies the DNS
fr	Found 2 DS records for fr in the . zone Found 1 RRSIGs over DS RRset RRSIG=46551 and DNSKEY=46551 verifies the DS RRs Found 4 DNSKEY records for fr DS=9392/SHA-256 verifies DNSKEY=9392/SEP Found 2 RRSIGs over DNSKEY RRset RRSIG=9392 and DNSKEY=9392/SEP verifies the DNSKEY RRset
univ-grenoble-alpes.fr	No DS records found for univ-grenoble-alpes.fr in the fr zone No DNSKEY records found univ-grenoble-alpes.fr A RR has value 195.83.24.194 No RRSIGs found

The parent zone data should include DS records for the child zone.

To remedy, the signer of the univ-grenoble-alpes.fr zone should send the current DS records to the fr zone.

[dnssec-analyzer.verisignlabs.com]

PKI applications: Web security

1. Respective situations of security systems in the protocols realm
2. SIGMA: Authenticated Diffie-Hellman
3. IPSec
4. SSL/TLS
   - OpenSSL
5. E-mail
   - S/MIME
   - PGP and GnuPG
   - DNSsec
   - OTR
6. Secure payments
   - SET: Secure Electronic Transactions
   - 3D-Secure
   - e-IDAS
   - Bitcoin
7. LDAP

OTR (Off-the-record)

- To instant messaging protocols (via AES, D-H, SHA …)
  - Confidentiality
  - Authentication
  - Deniability !
  - Perfect forward secrecy
- High level overview
  1. Alice signals to Bob that she would like (using an OTR Query Message) or is willing (using a whitespace-tagged plaintext message) to use OTR to communicate. Either mechanism should convey the version(s) of OTR that Alice is willing to use.
  2. Bob initiates the authenticated key exchange (AKE) with Alice. Version 2 of OTR uses a variant of the SIGMA protocol as its AKE.
  3. Alice and Bob exchange Data Messages to send information to each other.

SIGMA-R protocol as AKE in OTR

Alice | Bob

1) Random r and x
2) $AES_r(g^x)||h(g^x)$ -------->
 3) Random y
<-------- 4) g^y
5) $s=(g^y)^x$
6) $c,c',m_1,m_{1'},m_2,m_{2'}$ derived from s
7) $M_A=MAC_{m1}(g^x,g^y,Kpub_A,ID_A)$
8) $X_A=Kpub_A,ID_A,SIG_{KprivA}(M_A)$
9) $r||AES_c(X_A)||MAC_{m2}(AES_c(X_A))$ -------->
 10) $s,c,c',m_1,m_{1'},m_2,m_{2'}$
 11) Checks X_A, M_A
 12) $M_B=MAC_{m1'}(g^x,g^y,Kpub_B,ID_B)$
 13) $X_B=Kpub_B,ID_B,SIG_{KprivB}(M_B)$
<-------- 14) $AES_{c'}(X_B)||MAC_{m2'}(AES_{c'}(X_B))$
15) Checks X_B, M_B

OTR exchange

Alice | Bob

1) Random x_1
2) g^{x1} || $SIG_{KprivA}(g^{x1})$ -------->
 3) Random y_1
 4) $K_{11}= g^{x1y1}$
<-------- 4) g^{y1} || $SIG_{KprivB}(g^{y1})$
5) $K_{11}= g^{x1y1}$
6) Message M_1, Random x_2, $K_{21}= g^{x2y1}$
7) $C_1 = g^{x2}$ || $AES_{K11}(M_1)$
8) C_1 || $HMAC_{H(K11)}(C_1)$ -------->
 9) $K_{21}= g^{x2y1}$
 10) Message M_2, Random y_2, $K_{22}= g^{x2y2}$
 11) $C_2 = g^{y2}$ || $AES_{K21}(M_2)$
<-------- 12) C_2 || $HMAC_{H(K21)}(C_2)$
13) $K_{22}= g^{x2y2}$
14) Message M_3, Random x_3, $K_{32}= g^{x3y2}$
…

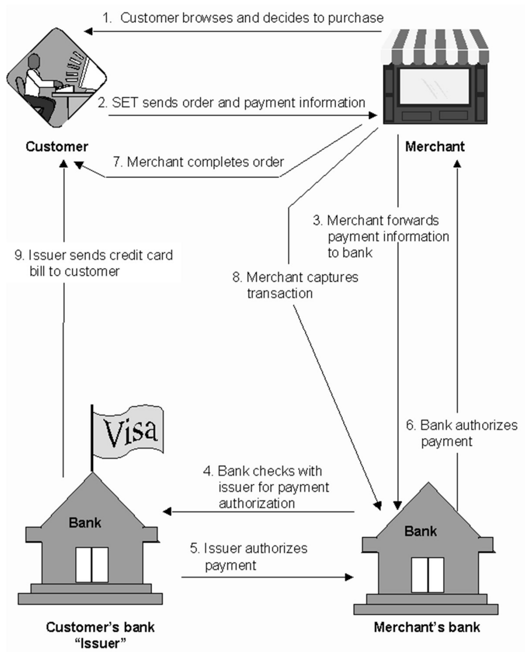
Bob cannot prove that Alice generated the MAC: He could have done it, too (Anyone who can verify can also forge)

Signatures authenticate the shared secret, not content

PKI applications: Web security

1. Respective situations of security systems in the protocols realm
2. SIGMA: Authenticated Diffie-Hellman
3. IPSec
4. SSL/TLS
   – OpenSSL
5. E-mail
   – S/MIME
   – PGP and GnuPG
   – DNSsec
   – OTR
6. Secure payments
   – SET: Secure Electronic Transactions
   – 3D-Secure
   – e-IDAS
   – Bitcoin
7. LDAP

Secure Electronic Transactions



- Visa security
  – Visa card no.
    - Art L 132-4: cardholder is not held liable in the event of fraudulent remote payment
    - Art L 132-5: Bank has one month to reimburse; no bank charges; cancellation lead time: 70 days min. to 120 max.
  – Expiration date
  – e-Visa
- MasterCard + Visa 1996
  – I3M, MS, Netscape, Verisign
  – Secure communications channel
  – Trust via X509
  – Confidentiality
  – Dual signature

SET: dual signature

- Couple payment instruction and order instruction
- $SD = E_{KprivAlice}(\ H(\ H(IP)\ ||\ H(IC)\ )\ )$

Alice $\quad$ ----- $E_{K1}(hip\ ||\ IC\ ||SD)$ ----> Trader

| $\qquad$ verification

| $\qquad$ $H(\ hip\ ||\ H(IC)\ ) =?\ D_{pubAlice}(SD)$

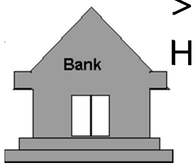
|

$E_{K2}(IP||hic||SD)$

|

\> Bank: verification

$H(\ H(IP)\ ||\ hic\ ) =?\ D_{pubAlice}(SD)$



SET: a PKI

⇨ Certificates required for dual signatures

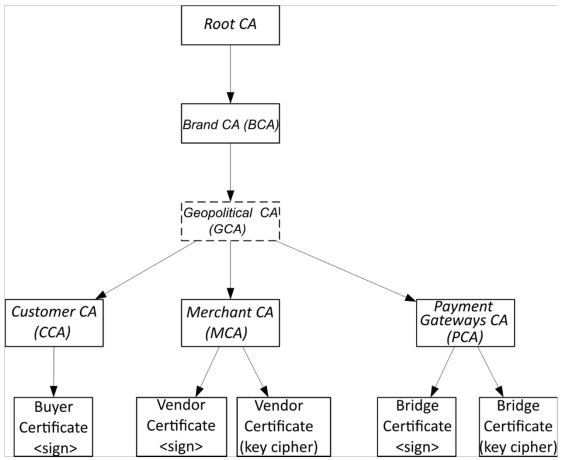
⇨ Interchange symmetrical secret keys between parties

⇨ Mutual authentications

⇨ Transactions:
- holder registration; trader registration;
- purchase order; payment authorization; payment acquisition;
- certificate status; purchase inquiry;
- authorization cancellation; acquisition cancellation;
- credit; credit cancellation;
- certificate request, etc.

SET PKI security architecture

- Root CA: EIG
- BCA: Visa/MasterCard, …
- GCA: country level
- CCA: buyers' bank or org.
- MCA: vendors' bank or org.
- PCA: interbanking system



3D-Secure ?

- 3 domains :
  - Seller bank (Acquirer Domain) : Merchant plug-in (MPI)
  - Buyer bank (Issuer Domain) : Access Control Server (ACS)
  - Credit-card system (Interoperabiblity Domain)

Client (pin, etc.) --→ Merchant

| ↑ $\qquad$ |

| | client $\qquad$ |

| | authentication: $\qquad$ |

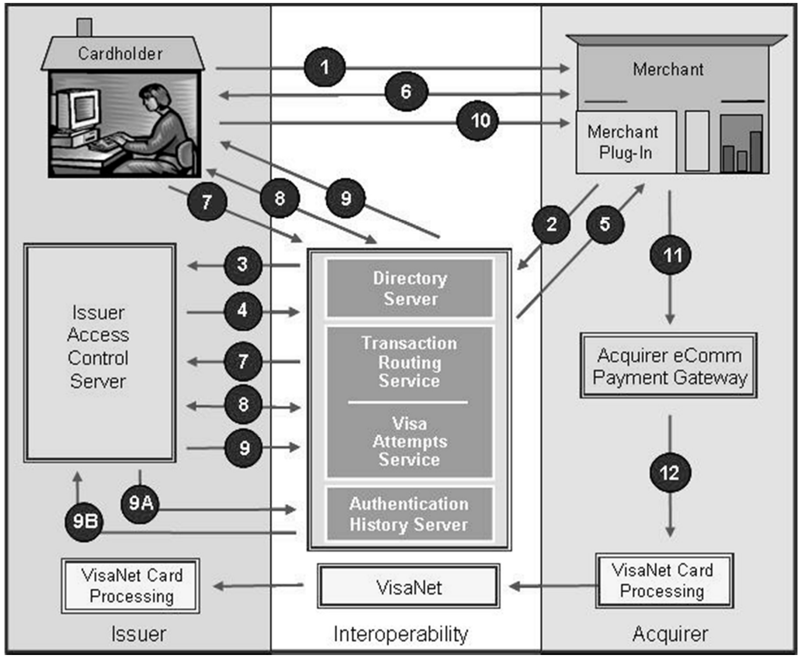
↓ | SMS, one-time-credit-card,… $\qquad$ ↓

Buyer Bank ←----------------------MPI-------------------→ Seller Bank

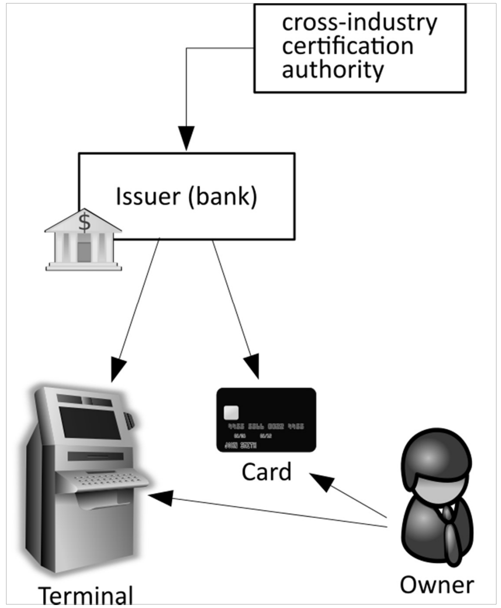
△ Phishing

△ Reverse onus (shifting the burden of proof on to the individual specified to disprove an element of the information …) ?

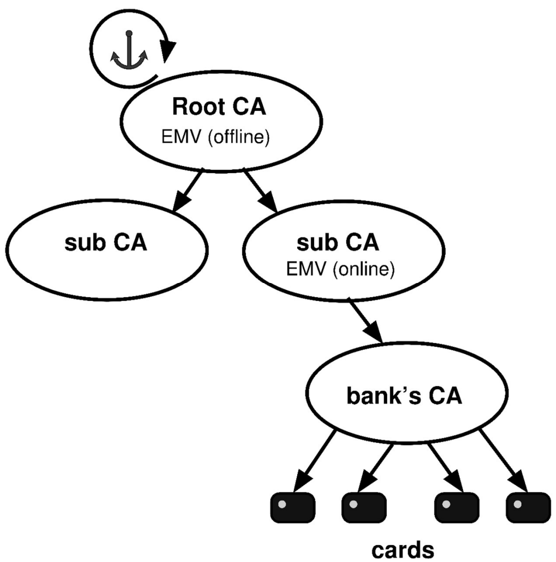
3D-Secure Interoperability domain



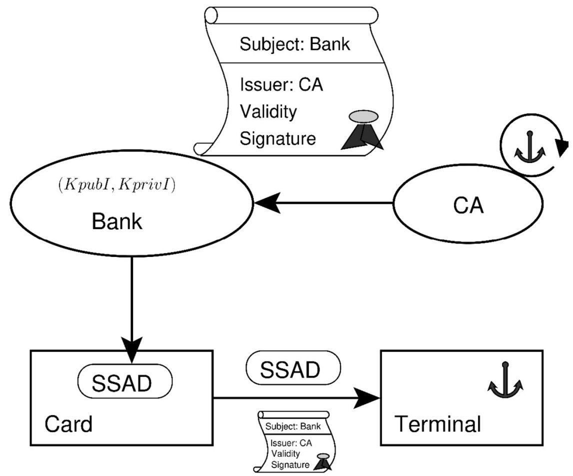
EMV (Europay, Mastercard, Visa)



EMV's PKI

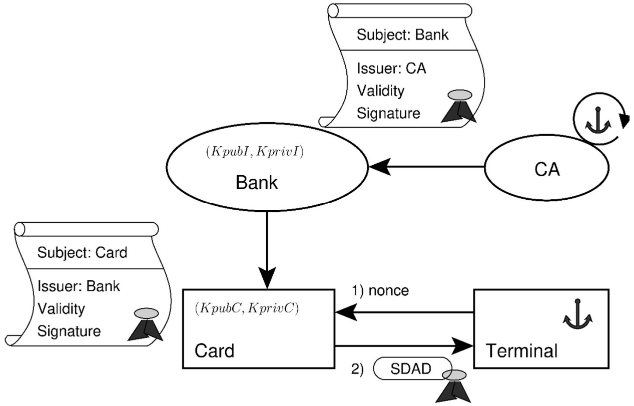


EMV static authentication (SDA)



- SSAD: Signed Static Application Data
  - Signed by the bank
  - Loaded into the card
- EMV's CA:
  - Bank's certificate issuer
  - Terminal's trust anchor
- Terminal:
  - checks certificate
  - Checks SSAD signature

EMV dynamic authentication



- SDAD: Signed Dynamic Application Data
  - Signed by the card
  - $SIG_C(nonce||date||\ldots)$
- EMV's CA:
  - Terminal's trust anchor
- Terminal:
  - Checks certificate chain

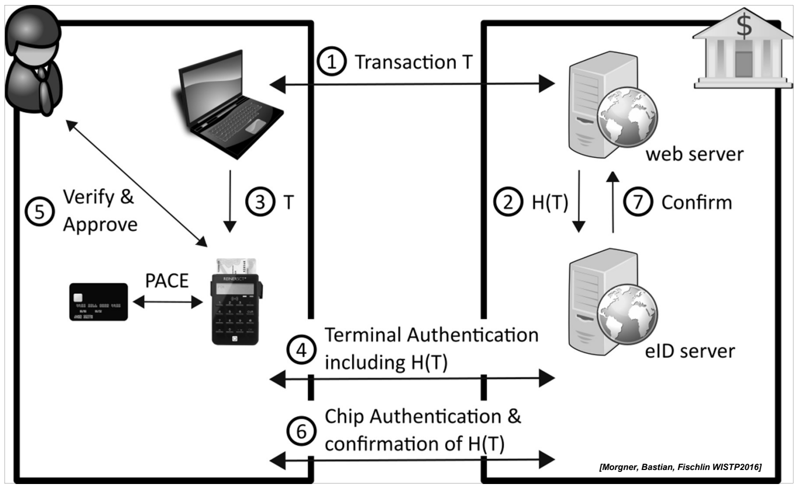
EMV CDA: combined DDA/Application Cryptogram Generation

- DDA:
  - $SDAD=SIG_C(nonce||date||\ldots)$
- CDA:
  - $SDAD=SIG_C(nonce_C||date||Hash(Transaction)||nonce_T||\ldots)$

PIN check

Chip		Terminal
DDA or CDA	$\longleftrightarrow$	Check $Cert_C$
nonce r	$\longrightarrow$	
	$\longleftarrow$	$m=E_{pubC}(PIN,r)$
$(PIN,r) = D_{privC}(m)$: OK	$\longrightarrow$	
$(PIN,r) \neq D_{privC}(m)$: CANCEL	$\longrightarrow$	

e-IDAS



- PACE: password-authenticated connection establishment

$$z=E_{H(\pi)}(g) \quad \longrightarrow \quad g=D_{H(\pi)}(z)$$
$$DH(K=g^{ab}) \quad \longleftrightarrow \quad DH(K=g^{ab})$$
$$PoK(K) \quad \longleftrightarrow \quad PoK(K)$$

(4) e-IDAS Transaction Authentication

- Fresh DH parameters
- Secure reception of H(T) from the web server
- Signature of H(T) and DH guaranteed by the PKI

Chip		eID server (via the terminal)
	$\longleftarrow$	$Cert_T$ (terminal certificate for $pub_T, priv_T$)
Check $Cert_T$		
	$\longleftarrow$	$f=F(g^v)$
nonce r	$\longrightarrow$	
	$\longleftarrow$	$\sigma = SIGN_{privT}(r,f,ID_E,H(T),\ldots)$
Check σ		

(6) e-IDAS Chip Authentication

- Signed acknowledgement on validation of H(T) by user

Chip		eID server (via the terminal)
$Cert_C$	----→	
		Check $Cert_C$
		Extract g^u = pub-key(Chip)
	←----	g^v
Check g^v with f		
nonce r'	----→	
$K=(g^v)^u$		$K=(g^u)^v$
$\tau = MAC_K(r', g^v)$	----→	
		Check τ
		☝ reports success to web server …

Bitcoin

1	BTC	1 bitcoin	
0.01	BTC	1 cBTC	1 bitcent
0.001	BTC	1 mBTC	1 millibitcoin
0.000 001	BTC	1 μBTC	1 microbitcoin
0.000 000 01	BTC	1 satoshi	

1. Unit of account
2. Eases exchange transactions
3. Storage of value
4. Not a legal currency
   - △ It can be refused
   - △ No guaranty of reimbursement

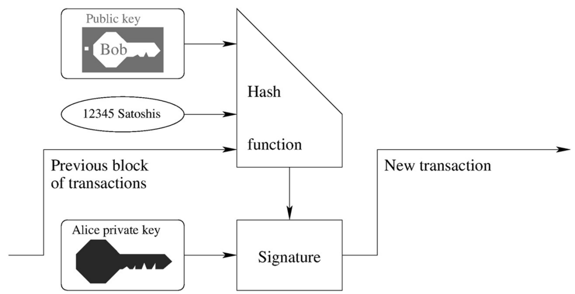
Main Characteristics

1. All the transactions are public
2. Transactions between public keys
   - ⇨ Private only while they stay in bitcoins
3. Peer-2-Peer guaranty of Validity
4. Adaptive Security / power of the world network

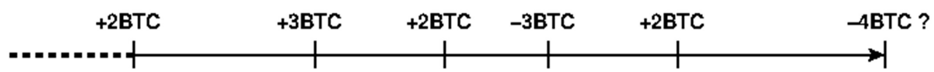
Bitcoin account number:

RIPEMD-160 (SHA-256 (SHA256 ($ECDSA_{pub}$)))

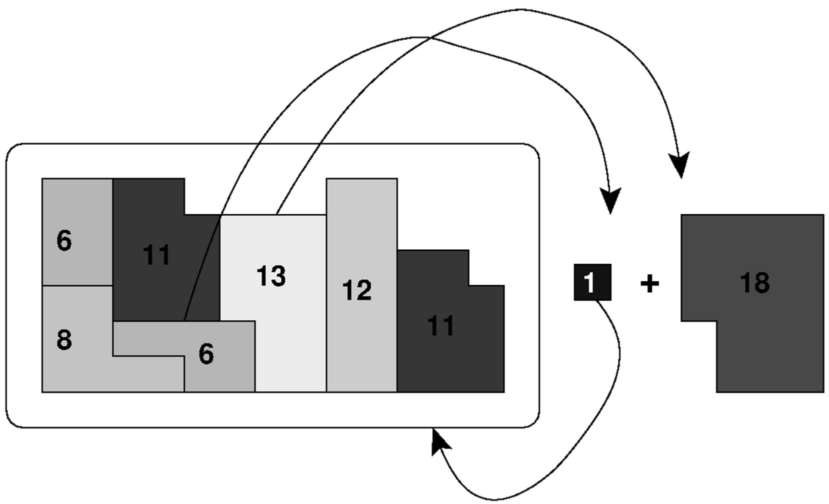
bitcoin Transaction



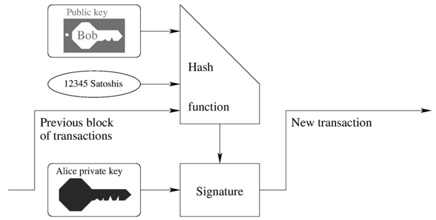
- Alice signs the public key of the new owner
- The previous transaction block is associated
- The (global) chain proves possession (by Alice)



Bitcoin wallet for a transaction of 18 bitcoins



Double spending?



- Alice signs the same block for several recipients?

⇨ All transactions are published

⇨ Transaction valid only after introduction in the chain

⇨ To insert in the chain one needs:

  ⇨ That Alice owns enough bitcoins

  ⇨ Find a valid hash for the transaction

  SHA-256 (SHA-256 (transaction block)) < target

  On average 2^{32}.Difficulty hashes to realize this

Mining

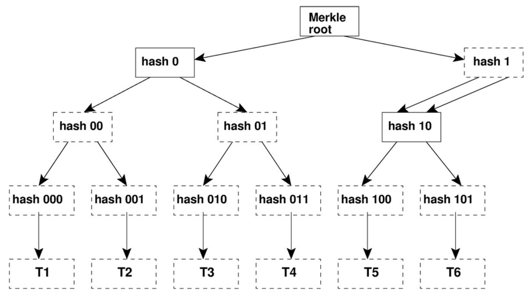
- Example of hash target:

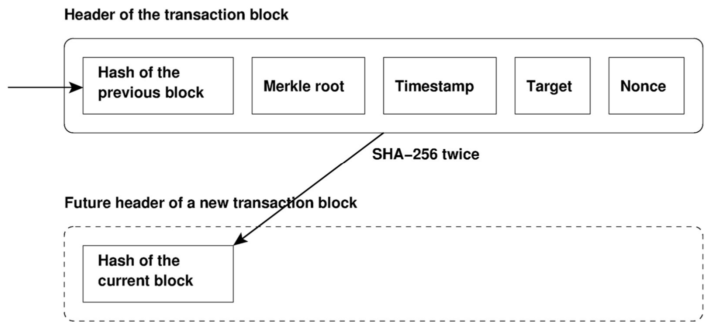
00000000000000002815EE000000000000000000000000000000000000000000

```
Nonce := 0;
repeat
    hashPrevBlock := last block validated by the network;
    Recover all transactions not yet validated;
    hashMerkleRoot := hash of the transactions to validate;
    Time := in seconds;
    Target := current hash target;
    Nonce := Nonce + 1;
    header :=(Version||hashPrevBlock||hashMerkleRoot||Time||Bits||Nonce)
until SHA-256(SHA-256(header)) < Target
```

Size of the world chain?

- Merkel hash tree for the transactions





- Chaining for blocks

Mining, transaction fees

- Mining should never stop, at a worldwide level
  - Otherwise transactions are not validated anymore
- Mining costs (computing resources)
  - Reward for discoverers of new blocks
  - Bitcoin creation, or transaction fees

- Reward, today (next halving 2017): 25 BTC / discovery
  - Only way to create bitcoins
- Reward divided by 2 every 210000 validated blocks
  - From 50 BTC to 1 satoshi:

  Maximum $\sum 50/2^i$ § 210000 = 21 millions of bitcoins

Privacy, monetary usage

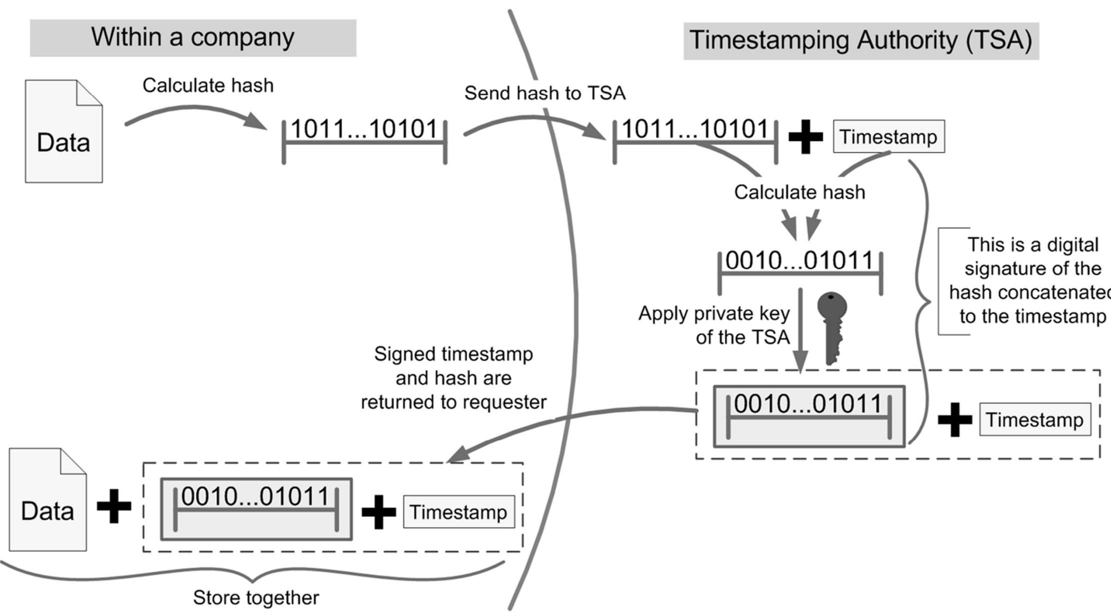
- All transactions are public, but signed
  - Protection of the Identity of private key's owners

- Privacy is removed for the whole set of transactions of an owner:
  - Only when exchanging with other currencies
  - At least to the exchange institution

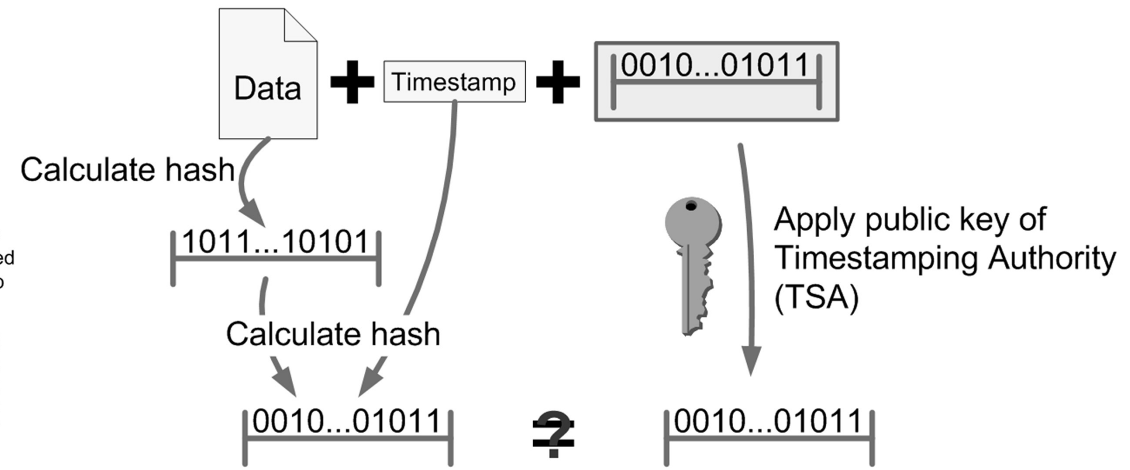
- The number of bitcoins is limited
  - Largely used: increasing value, speculating interest
  - Savings interest: scarcely used, decreasing value

- Success?
  - Bitcoin transactions will be taxed …
  - If private entities owns a majority of the mining power, they become banks, de facto

Trusted timestamping



Checking the trusted timestamp



If the calculated hashcode equals the result of the decrypted signature, neither the document or the timestamp was changed and the timestamp was issued by the TTP. If not, either of the previous statements is not true.

Cf. Electronic postmark (ex: <u>certinomis</u>)

PKI applications: Web security

1. Respective situations of security systems in the protocols realm
2. SIGMA: Authenticated Diffie-Hellman
3. IPSec
4. SSL/TLS
   - OpenSSL
5. E-mail
   - S/MIME
   - PGP and GnuPG
   - DNSsec
   - OTR
6. Secure payments
   - SET: Secure Electronic Transactions
   - 3D-Secure
   - e-IDAS
   - Bitcoin
7. LDAP

LDAP *[RFC2255]*

- LDAP is a standard and extendible directory protocol. It provides:
  - the *protocol* giving access to the information contained in the directory
  - an *information model* defining the type of data contained in the directory
  - a *naming model* defining how the information is organized and referenced
  - a *functional model* that defines how to access the information
  - a *security model* that defines how data and access are protected
  - a *duplication model* that defines how the database is distributed between servers
  - *APIs* for developing client applications
  - *LDIF*, a data interchange format

LDAP information

- An entry of person type is represented in the following mannner:
  - dn: cn= June Rossi, ou= accounting, o= Ace Industry, c= US
  - objectClass: person
  - objectClass: organizationalPerson
  - objectClass: inetOrgPerson
  - cn: June Rossi
  - sn: Rossi
  - givenName: June
  - mail: rossi@aceindustry.com
  - userPassword: {sha}KDIE3AL9DK
  - uid: rossi
  - telephoneNumber: 2616
  - roomNumber: 220

Examples of LDAP applications

- LDAP can provide different services. You can use it as:
  - A data dissemination protocol: LDAP directory like a telephone book to contact people
  - A service protocol for applications: LDAP directory of mail addresses that allow mail servers or tools to compose or check addresses, used by list managers (Sympa), database for storing an application's configuration settings (Communicator profiles)
  - A gateway between applications: allowing the interchanging of data between incompatible applications – Netscape Communicator and Microsoft Outlook address books, for example
  - A service protocol for operating systems or Internet services: LDAP database of users employed by systems for handling authentication (authentication certificates) and managing rights of access to network resources – projects are in progress for replacing NIS and NIS+ with LDAP

LDAP authentication

- LDAP offers several authentication choices:
  - *Anonymous authentification* – corresponds to access to a server without authentication, which only makes it possible to consult data that are available for reading by anyone
  - *Root DN Authentication* – this is the privileged user, who has access to all data
  - *Password in clear* – this is the conventional method by which the password travels across the network in clear. Suitable for a switched network or a network behind a firewall and for non-sensitive data
  - *Password + SSL or TLS* – the session between the server and the client is encrypted, and the password no longer travels in clear
  - *Certificates + SSL*
  - *Simple Authentification and Security Layer* (SASL) – makes it possible to employ key-based mechanisms (OTP, Kerberos, etc.)

OpenLDAP

- The LDAP server needs copies of its own certificate and key files in addition to the CA certificate in X509 format (use e.g. openssl)

  - The key file must only be readable by the server process, so make it mode 400 and owned by the owner of the server (OpenLDAP can be run as root or be told to set UID to another user after binding the TCP ports)

  - Add lines to *slapd.conf* to tell the server about the new files:
    - TLSCertificateFile /etc/openldap/keys/server.cert
    - TLSCertificateKeyFile /etc/openldap/keys/server.key
    - TLSCACertificateFile /etc/openldap/keys/ca.cert

LDAP & LDAPS URL

- ldap://host:port/dn?attributes?scope?filter?extensions
  - The default hostname is localhost; the default port is 389. The default root distinguished name is the empty string. Authentication information may be specified in the extensions portion of the URL

- LDAPS == LDAP + SSL
  - On connection, the client and the server establish a TLS link before any other LDAP command is sent (without sending LDAP's *StartTLS* command)
  - The LDAPS link must be closed before the closure of TLS (whereas with *StartTLS*, it is possible to switch from a secure link to an unsecured link and vice versa)

  - ldaps://host:port/dn?attributes?scope?filter?extensions
    - Default port for LDAPS URLs is 636 instead of 389