

Hybrid Encryption and Further Topics

Cryptography (lecture portion)

Andreas Unterweger

School of ITS
Salzburg UAS

Winter term 2024/25

- Symmetric cryptography
 - Key exchange
 - DH(E)
 - AES
- Asymmetric cryptography
 - Public key
 - Private/secret key
 - RSA
- Cryptographic hashes
 - Collision resistance
 - (Second) pre-image resistance
 - SHA-2

- Hybrid encryption
 - Key exchange revisited (with key derivation)
 - Hybrid encryption scheme
 - Challenge-response scheme
- Examples of combined use of cryptography
 - Secure password storage
 - Transport Layer Security
 - Practical protocols (choice)

Key exchange revisited I

- Diffie-Hellman key exchange
 - Alice and Bob exponentiate random numbers in a modulus (multiplicative group)
 - Rules for exponents give Alice and Bob the same group element
 - Final group element can be used to derive a shared key
 - Eve cannot feasibly compute the shared key (discrete logarithm problem and Diffie-Hellman assumptions)
- Challenges
 - Choosing safe groups
 - Converting the final group element into a key \rightarrow key derivation

Key exchange revisited II

- Key derivation
 - Derive a key k from the final group element f
 - Example: Get 128-bit AES key from 1,024-bit group element
 - Multiple possibilities
- Hashing for key derivation
 - Compute a cryptographic hash: $k := H(f)$
 - Removes any remaining algebraic structure
 - Yields a fixed-size result (key)
- Standardized advanced versions, e.g., PBKDF2
 - Repeated use of hashes, e.g., 1,000 iterations
 - Slows down attackers significantly
 - Use of additional salt (details later)

[1] Kaliski, B.: PKCS #5: Password-Based Cryptography Specification Version 2.0. <https://www.ietf.org/rfc/rfc2898.txt> (accessed on October 8, 2022), 2000.

Hybrid encryption I

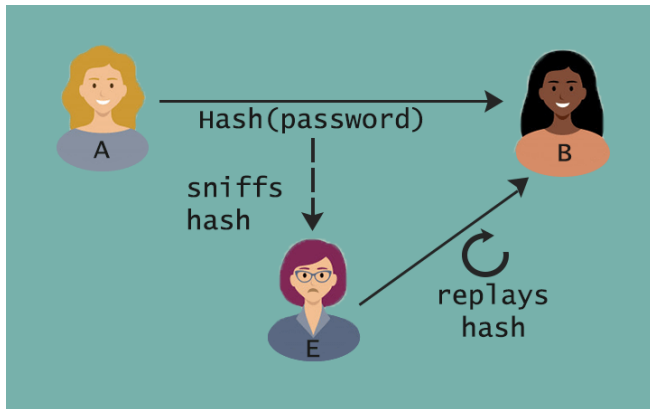
- Alternative to Diffie-Hellman key exchange
- Combine asymmetric and symmetric ciphers
- Assumptions
 - Bob has a public-private key pair $(k_{public}^A, k_{private}^A)$
 - Alice knows Bob's public key (recall certificates!)
- Key exchange steps:
 - 1 Alice generates a random key k
 - 2 Alice encrypts k with Bob's public key: $k_e = E(k_{public}^B, k)$
 - 3 Alice sends the encrypted key k_e to Bob
 - 4 Bob decrypts the received key with his secret key: $k = D(k_{private}^B, k_e)$
 - 5 Alice and Bob can exchange messages with symmetric key k

Hybrid encryption II

- Why not use asymmetric cryptography for all messages?
 - Public key cryptography is much less efficient than secret key cryptography (longer messages require much more expensive computations)
 - RSA keys are much longer than AES keys for the same level of security (recall elliptic curves for a viable alternative!)
- Summary of hybrid encryption
 - Use public key cryptography to exchange the key
 - Use secret key cryptography to exchange messages
 - The symmetric key is practically random
 - Bob's public key must be known in advance for this to work
- Supplement to this scheme: Challenge-response scheme

Challenge and response I

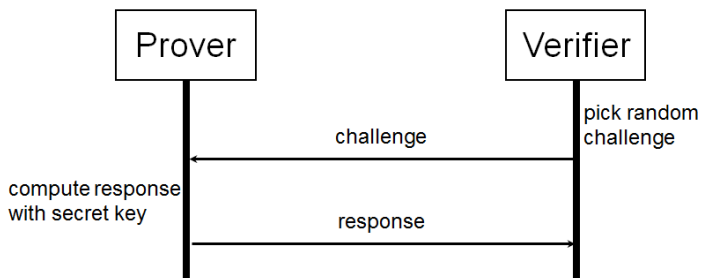
- How to verify that Bob is talking to Alice and not to Eve?
- Risk of replay attacks by Eve



Source: Gibson, D.: Replay Attacks. <https://cybersecurityglossary.com/replay-attacks/> (accessed on October 8, 2022), 2020.

Challenge and response II

- How to solve authentication?
 - Certificates (lack liveness, allow for replay attacks)
 - Challenge and response (impedes replay attacks by testing liveness)



Source: Rathgeb, C.: IT-Sicherheit, Kapitel 6 – Authentifikation.

https://www.dasec.h-da.de/wp-content/uploads/2014/05/06_authentifikation.pdf (accessed on October 8, 2022), 2014.

[2] Barrett, D. J. and Silverman, R. E.: SSH: The Secure Shell: The Definitive Guide, 1st ed., O'Reilly, 2001.

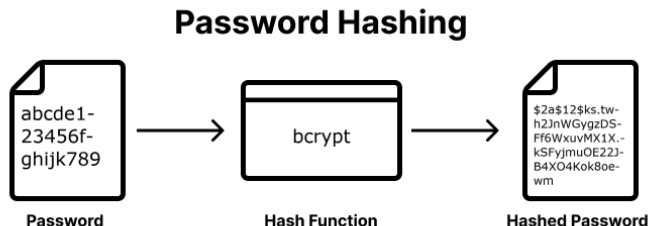
Challenge and response III

- Example: Challenge-response scheme for public-key authentication in the Secure Shell (SSH) protocol (simplified):
 - 1 Alice sends her public key k_{public} to Bob
 - 2 Bob generates a random challenge r and encrypts it with Alice's supposed public key: $c := E(k_{public}, r)$
 - 3 Bob sends the challenge c to Alice
 - 4 Alice decrypts the supposed challenge c' with her secret key k_{secret} :
 $r' = D(k_{secret}, c')$
 - 5 Alice hashes the decrypted challenge r' : $h' := H(r')$
 - 6 Alice sends the hashed response h' to Bob
 - 7 Bob hashes his random challenge r : $h := H(r)$
 - 8 Bob checks whether $h = h'$ (if not, he may be talking to Eve)
- Randomness impedes replay attacks
- Hashing is required so that chosen-plaintext attacks are avoided

[2] Barrett, D. J. and Silverman, R. E.: SSH: The Secure Shell: The Definitive Guide, 1st ed., O'Reilly, 2001.

Overview of secure password storage

- How to store (user) passwords?
 - Not as plaintext (obvious)
 - Encrypted? → Decryption key yields plaintext passwords
 - Hashed? → Identical passwords yield identical hashes
 - In practice: Salted and stretched

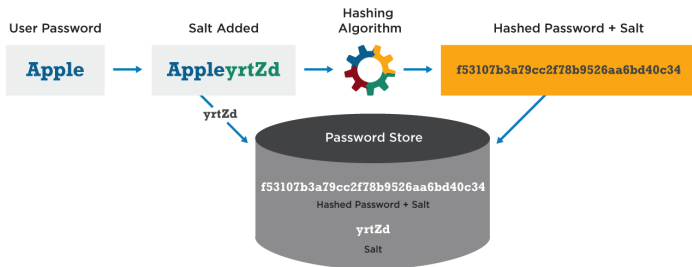


Source: Authgear: Password Hashing and Salting Explained. <https://www.authgear.com/post/password-hashing-salting> (accessed on October 8, 2022), 2022.

[3] Arias, D.: Adding Salt to Hashing: A Better Way to Store Passwords. <https://auth0.com/blog/adding-salt-to-hashing-a-better-way-to-store-passwords/> (accessed on October 8, 2022), 2021.

- Salt: Random number (stored together with password) **per user**
- Salting: Concatenating the password p with a salt s before hashing
- Salted hashing: $\hat{H}(p, s) := H(p||s)$

Password Hash Salting

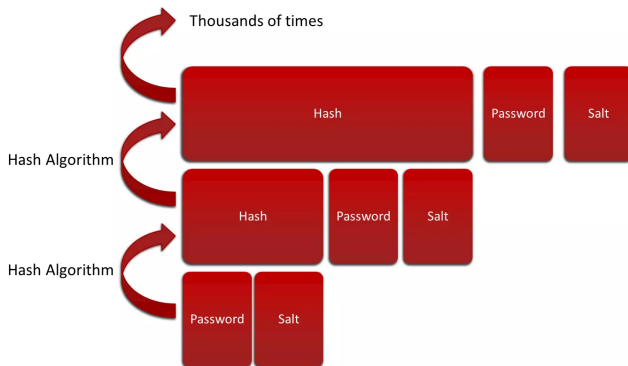


Source: Taylor, C.: Password Salting. <https://cyberhoot.com/cybrary/password-salting/> (accessed on October 8, 2022), 2020.

[3] Arias, D.: Adding Salt to Hashing: A Better Way to Store Passwords. <https://auth0.com/blog/adding-salt-to-hashing-a-better-way-to-store-passwords/> (accessed on October 8, 2022), 2021.

Stretching

- Stretching: Prolonging computation by repeated hashing
- Iterations: $P_1 = \hat{H}(p, s), P_{n>1} = \hat{H}(P_{n-1} || p, s)$
- Stored password hash: P_N for large N , e.g., 10,000







Source: Romero, M. I.: Risks and Challenges of Password Hashing.

<https://www.sitepoint.com/risks-challenges-password-hashing/> (accessed on October 8, 2022), 2014.

Attacks on securely stored passwords I

- Rainbow tables (pre-computed hashes when no salts are used)
- Dictionary attack (try a list of common passwords)
- Brute force (try all passwords from a defined set)

				
Password	p4s5w3rdz	p4s5w3rdz	p4s5w3rdz	p4s5w3rdz
Salt	-	-	et52ed	ye5sf8
Hash	f4c31aa	f4c31aa	1vn49sa	z32i6t0

Adopted from Arias, D.: Adding Salt to Hashing: A Better Way to Store Passwords.

<https://auth0.com/blog/adding-salt-to-hashing-a-better-way-to-store-passwords/> (accessed on October 8, 2022), 2021.

[3] Arias, D.: Adding Salt to Hashing: A Better Way to Store Passwords.

<https://auth0.com/blog/adding-salt-to-hashing-a-better-way-to-store-passwords/> (accessed on October 8, 2022), 2021.

Attacks on securely stored passwords II

- Brute force attacks depend on the assumed password alphabet
 - Example: 8 character-passwords with English alphabet (only upper and lower case): Key space is
$$52^8 > 50^8 = \left(\frac{100}{2}\right)^8 = \frac{(10^2)^8}{2^8} = \frac{10^{16}}{2^8} = 2^2 \cdot \frac{10^{16}}{2^{10}} \approx 4 \cdot \frac{10^{16}}{10^3} = 4 \cdot 10^{13}$$
 - Dictionary attacks are only successful when the password is contained in the dictionary (otherwise brute force attack required)
 - Stretching can slow down dictionary **and** brute force attacks
 - Salting makes rainbow tables futile
- Use salting and stretching (better: standardized algorithms)
- User recommendation: Use a password manager

[3] Arias, D.: Adding Salt to Hashing: A Better Way to Store Passwords.

<https://auth0.com/blog/adding-salt-to-hashing-a-better-way-to-store-passwords/> (accessed on October 8, 2022), 2021.

Overview of Transport Layer Security (TLS)

- Based on the now-obsolete Secure Sockets Layer (SSL)
- Protocol to provide a secure channel between Alice and Bob
- Guarantees
 - Authentication (Bob is always authenticated, Alice optionally)
 - Confidentiality (only Alice and Bob can read data, but not an attacker)
 - Integrity (data cannot be changed by an attacker without detection)
- Uses a variety of cryptographic primitives, e.g.,
 - Asymmetric ciphers
 - Symmetric ciphers
 - Cryptographic hashes
 - Digital signatures
 - Digital certificates

[4] Rescorla, E.: The Transport Layer Security (TLS) Protocol Version 1.3 (RFC 8446).
<https://www.rfc-editor.org/rfc/rfc8446.html> (accessed on October 26, 2022), 2018.

Recap: Digital signatures

- Generated and added to a message by the signer/sender
- Can be checked by the receiver (and everybody else)
- Preserve message integrity instead of message privacy
- Guarantees
 - The signed message has not been modified
 - The signed message originates from the signer
- Non-repudiation: A signer cannot claim **not** to have signed the message
- Rely on public key cryptography and hashes
 - Signature: Hash of message, encrypted with secret key
 - Signature verification: Compare hash and decrypted signature
 - If the signature check fails, either the message has been tampered with or someone else (a different key) has signed the message

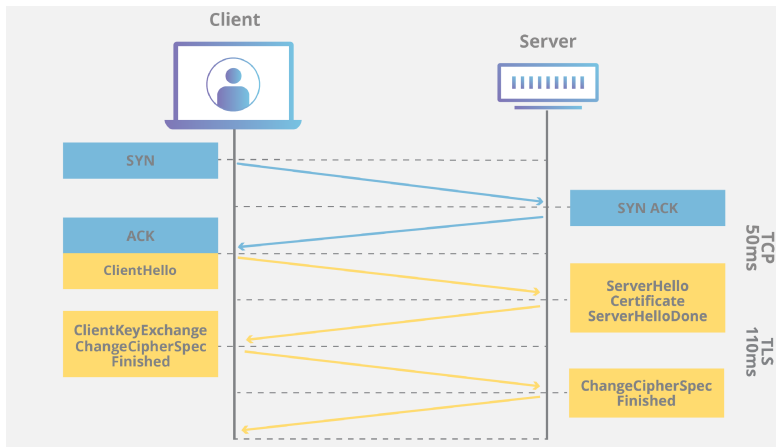
- Digital certificate
 - associates a public key with a person/entity
 - contains information about the signer, the person and their key
 - is signed by a trusted party
 - allows verifying signature and thus identity-to-key matching
 - Assumption: Trusted party whose public key is known by all parties
 - Certificate authorities (CAs)
 - are trusted to issue certificates
 - check the identity of the owner or entity
 - are assumed to have a known public key
 - Certificate authority hierarchies (CA hierarchies)
 - Issuing of certificates is delegated to sub-CA(s)
 - Multi-level certificates from CA to sub-CA(s) to entity
- Hierarchical trust model
- Root/top-level CA(s) self-sign their own certificate(s)

Key and cipher agreement I

- TLS relies on transport protocols such as TCP
- Initial handshake
 - Authenticates communicating parties through certificates
 - Negotiates used ciphers (not every party supports every cipher)
 - Performs key exchange with forward secrecy
 - Tamper-resistant even against active attackers (who modify messages)
- Used cryptographic primitives
 - Digital certificates for authentication (simplified)
 - Digital signatures for supported ciphers and certificates
 - (EC)DHE and similar protocols for key exchange
 - Hashes of the transcript (sequence of handshake messages)

[4] Rescorla, E.: The Transport Layer Security (TLS) Protocol Version 1.3 (RFC 8446).
<https://www.rfc-editor.org/rfc/rfc8446.html> (accessed on October 26, 2022), 2018.

Key and cipher agreement II



Source: Cloudflare: What is Transport Layer Security (TLS)?

<https://www.cloudflare.com/learning/ssl/transport-layer-security-tls/> (accessed on October 26, 2022), 2022.

[4] Rescorla, E.: The Transport Layer Security (TLS) Protocol Version 1.3 (RFC 8446).

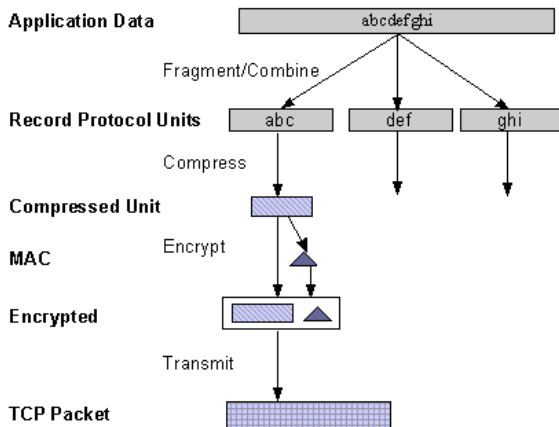
<https://www.rfc-editor.org/rfc/rfc8446.html> (accessed on October 26, 2022), 2018.

Secure message exchange I

- TLS can exchange messages after finished handshake
- Record protocol
 - 1 (Alice) Message is split into blocks
 - 2 (Alice) Blocks (records) are protected
 - 3 (Alice) Protected records are transmitted
 - 4 (Bob) Protected records are received
 - 5 (Bob) Protected records are checked and unprotected
 - 6 (Bob) Blocks (records) are reassembled into message
- Used cryptographic primitives
 - Symmetric block ciphers in selected modes (without details)
 - Message authentication codes (MACs, similar to hashes, no details)
- Optional compression step (without details)

[4] Rescorla, E.: The Transport Layer Security (TLS) Protocol Version 1.3 (RFC 8446).
<https://www.rfc-editor.org/rfc/rfc8446.html> (accessed on October 26, 2022), 2018.

Secure message exchange II



Source: The Apache Software Foundation: SSL/TLS Strong Encryption: An Introduction.
https://httpd.apache.org/docs/2.4/ssl/ssl_intro.html (accessed on October 26, 2022), 2022.

[4] Rescorla, E.: The Transport Layer Security (TLS) Protocol Version 1.3 (RFC 8446).
<https://www.rfc-editor.org/rfc/rfc8446.html> (accessed on October 26, 2022), 2018.

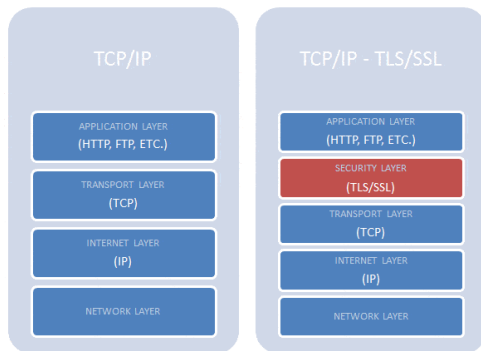
Practical concerns I

- Newer versions of TLS implement new ciphers and features
 - SSL and older TLS versions are
 - vulnerable to certain attacks (without details)
 - support broken ciphers/hash algorithms etc.
 - support weak algorithms and key lengths
- Deprecation or removal of flawed/weak algorithms and protocol parts
- Versioning (current: TLS 1.3)
- Byte-by-byte illustration of a TLS 1.3 connection (including handshake): <https://tls13.xargs.org/>

[4] Rescorla, E.: The Transport Layer Security (TLS) Protocol Version 1.3 (RFC 8446).
<https://www.rfc-editor.org/rfc/rfc8446.html> (accessed on October 26, 2022), 2018.

Practical concerns II

- TLS is designed to provide extra layer for application(-level) protocols



Source: Slaviero, M.: TLS/SSL and .NET Framework 4.0.

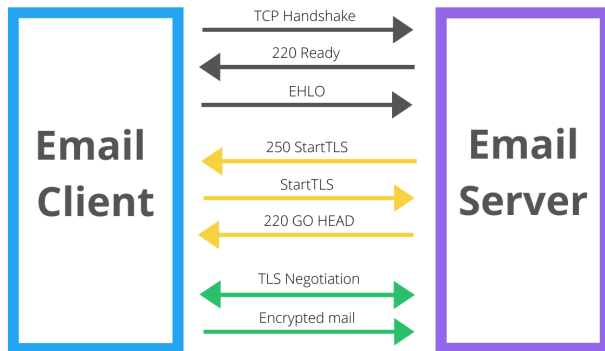
<https://www.red-gate.com/simple-talk/development/dotnet-development/tlssl-and-net-framework-4-0/> (accessed on October 26, 2022), 2011.

[4] Rescorla, E.: The Transport Layer Security (TLS) Protocol Version 1.3 (RFC 8446).

<https://www.rfc-editor.org/rfc/rfc8446.html> (accessed on October 26, 2022), 2018.

Practical concerns III

- Retrofitting unprotected protocols through STARTTLS command



Source: Griffin, J.: What is StartTLS? <https://sendgrid.com/blog/what-is-starttls/> (accessed on October 26, 2022), 2020.

[5] Griffin, J.: What is StartTLS? <https://sendgrid.com/blog/what-is-starttls/> (accessed on October 26, 2022), 2020.

Practical protocols (choice)

- Practical cryptographic protocols to illustrate combined use of cryptography (choose one):
 - Masking (smart meter data aggregation) [6]
 - Proof of work blockchains (electronic cash) [7]
 - Wi-Fi Protected Access 3 (WPA3, wireless communication) [8, 9]
 - Content protection (video streaming) [10, 11]
 - Further topics (suggestions)

[6] Kursawe, K., Danezis, G. and Kohlweiss, M.: Privacy-friendly Aggregation for the Smart Grid. In PETS 2011: Privacy Enhanced Technology Symposium, pp. 175–191, 2011.

[7] Nakamoto, S.: Bitcoin: A Peer-to-Peer Electronic Cash System. <https://bitcoin.org/bitcoin.pdf> (accessed on October 26, 2022), 2008.

[8] WiFi Alliance: WPA3™ Specification Version 3.3. <https://www.wi-fi.org/file/wpa3-specification> (accessed on July 10, 2024), 2024.

[9] WiFi Alliance: Wi-Fi Protected Access® Security Considerations.

https://www.wi-fi.org/system/files/Security_Considerations_20210511.pdf (accessed on July 10, 2024), 2021.

[10] European Telecommunications Standards Institute and European Broadcasting Union: Digital Video Broadcasting (DVB); Content Protection and Copy Management (DVB-CPCM); Part 2: CPCM Reference Model (ETSI TS 102 825-2 V1.2.1), 2011.

[11] European Telecommunications Standards Institute and European Broadcasting Union: Digital Video Broadcasting (DVB); Content Protection and Copy Management (DVB-CPCM); Part 5: CPCM Security Toolbox (ETSI TS 102 825-5 V1.2.1), 2011.

Thank you for your attention!

Questions?