Nuts and Bolts of Encryption: A Primer for Policymakers

Edward W. Felten

Center for Information Technology Policy
Department of Computer Science
Woodrow Wilson School of Public and International Affairs
Princeton University

version of April 26, 2017[1]

This paper offers a straightforward introduction to encryption, as it is implemented in modern systems, at a level of detail suitable for policy discussions. No prior background on encryption or data security is assumed.

Encryption is used in two main scenarios. *Encrypted storage* allows data to be stored on a device, with encryption protecting the data should a malicious party get access to the device. *Encrypted communication* allows data to be transmitted from one party to another party, often across a network, with encryption protecting the data should a malicious party get access to the data while it is in transit. Encryption is used somewhat differently in these two scenarios, so it makes sense to present them separately. We'll discuss encrypted storage first, because it is simpler.

We emphasize that the approaches described here are not detailed descriptions of any particular existing system, but rather generic descriptions of how state-of-the-art systems typically operate. Specific products and standards fill in the details differently, but they are roughly similar at the level of detail given here.

Encrypted storage

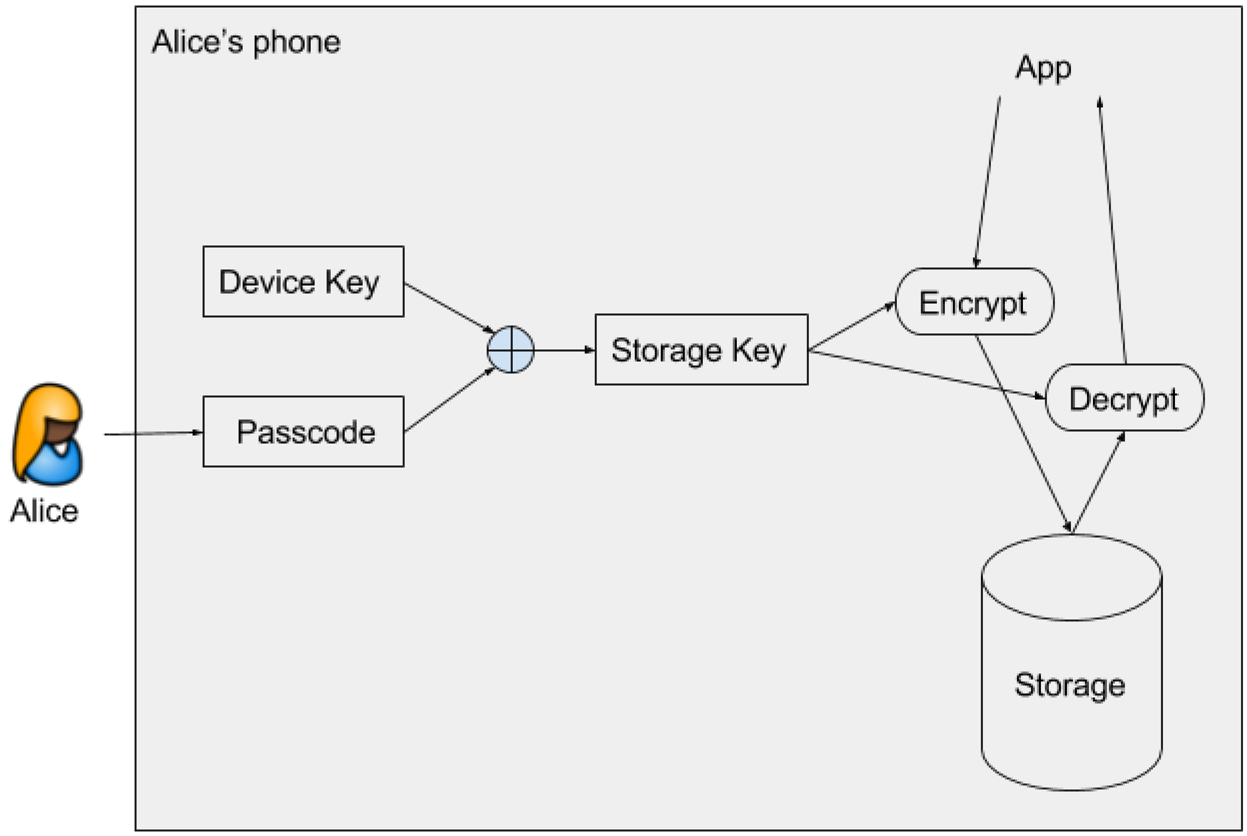
Suppose a user, Alice, wants to store data on a device, which might be a smartphone in her possession, or might be a storage server operated by a service provider. Alice generates a secret key that only she knows, and she uses the secret key to encrypt the data. Encryption protects the *confidentiality* of the data, so that a malicious party who gets access to the device, but does not know the secret key, cannot learn the contents of Alice's data. Encryption also protects the *integrity* of the data, so that a malicious party who gets access to the device, but does not know the secret key, cannot tamper with the data without Alice detecting that tampering occurred.

Encryption on a device such as a smartphone typically works as depicted below. A device key, which is unique to Alice's specific phone, is built into the phone when the phone is

[1] This work is licensed under a Creative Commons Attribution 4.0 International License (https://creativecommons.org/licenses/by/4.0/). An up-to-date version of this paper will be available at https://www.cs.princeton.edu/~felten/encryption_primer.pdf

manufactured. In addition, Alice enters a secret passcode when she unlocks the phone. The device key and passcode are combined by cryptographic means to create a storage key, which will be used to encrypt data. From that point on, whenever an app wants to store data, the data is encrypted with the storage key, before the data is put into storage. Whenever an app wants to retrieve data from storage, the data is decrypted before it is returned to the app. When the system decrypts data, the system also checks for tampering.

Because all data is encrypted before it is put into storage, a malicious party who steals the device but does not know Alice's secret key cannot recover Alice's data, nor can such a party tamper with Alice's data without detection.



The security of data on the device depends ultimately on two keys. The use of the device key ensures that data can be decrypted only on Alice's specific phone—and the phone typically is physically "hardened" so that it is very difficult for a malicious party to extract the device key. The use of Alice's passcode ensures that Alice must take explicit action—entering her passcode—to enable decryption or authorized modification of data.

When Alice locks the phone, or when the phone loses power, Alice's passcode and the storage key are erased from the phone. At that point the phone no longer contains the key information that would enable stored information to be recovered or to be tampered with undetectably. Decryption and tampering are not possible because the storage key is not present. The storage key cannot be re-created because Alice's passcode is not present. Only by entering Alice's passcode can decryption and authorized modification of the data be made possible again.

But these protections will be in vain if a malicious party can guess Alice's passcode. In practice, users often choose passcodes that are easily guessable by a computer that can try a large number of guesses very quickly. A secure system must have additional defenses against password guessing. Typically this involves having the system impose a delay after a failed attempt to enter the passcode, and having the system stop accepting passcode attempts altogether after a certain number of failed attempts. This will make passcode guessing infeasible, unless Alice chooses an exceptionally weak passcode such as 0000 or her birthday.

Encrypted Communication

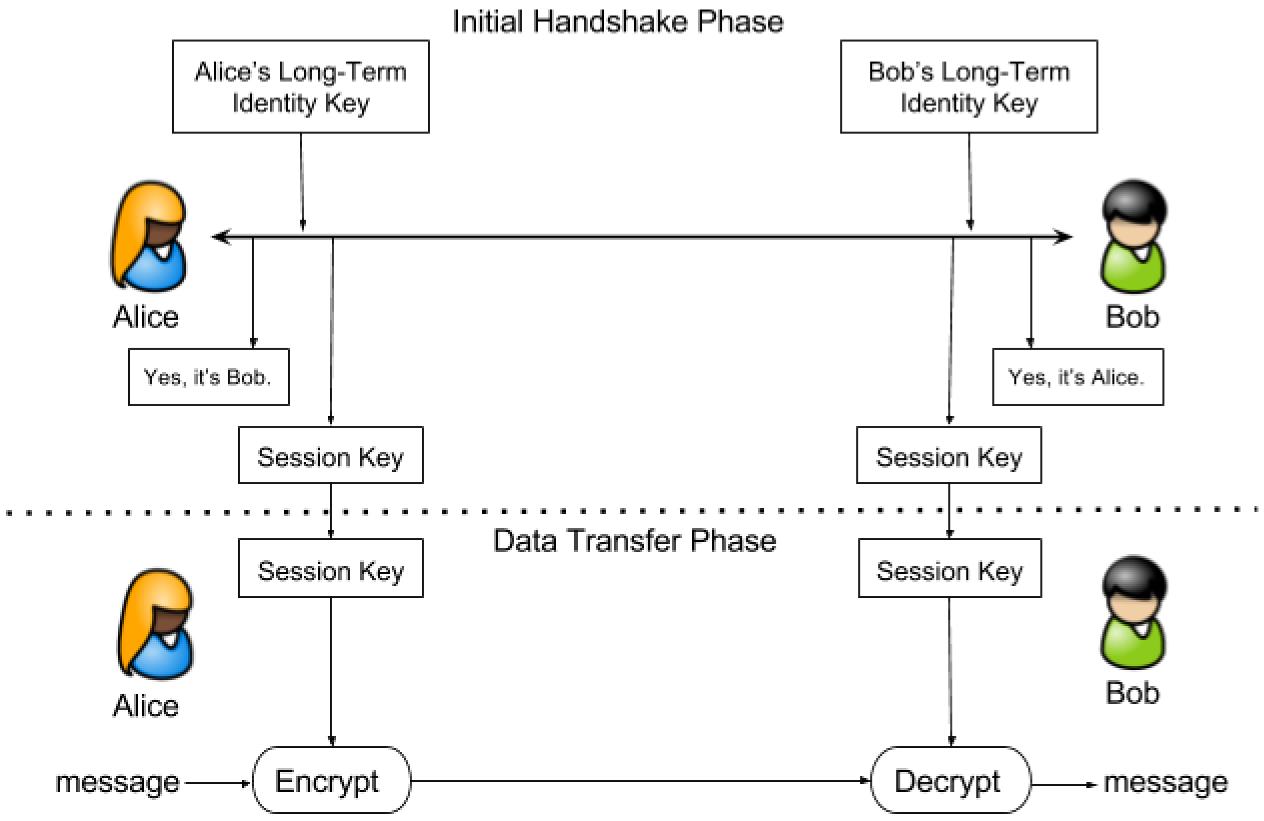
Encrypted communication works differently.

Suppose two users, Alice and Bob, want to send a series of messages to each other. They want to use encryption to protect the *confidentiality* of messages (so that nobody else can learn the contents of messages) and the *integrity* of messages (so that nobody else can tamper with messages without detection); and they want to use encryption to *authenticate* each other (so they both know they are not communicating with an impostor).

For encrypted communication, each party generates a *long-term identity key*, which they keep secret. A party can use its long-term identity key to prove its identity to other parties.

As depicted below, encrypted communication operates in two phases. In the first phase, the *handshake,* the two parties exchange a series of specially constructed messages. If anyone tampers with a message during the handshake, Alice and Bob will detect the tampering and abort the handshake. Otherwise, the handshake will succeed and will have two results: each party will get confirmation of the other's identity (i.e. that the other party is the real Alice or Bob, and not an impostor), and Alice and Bob will agree on a secret *session key* that is known only to the two of them. The details of how the initial handshake procedure gets these results are complex but not directly relevant to the policy discussion.

Having completed the initial handshake, Alice and Bob can proceed to send messages to each other. If Alice wants to send a message to Bob, she encrypts that message with the session key and sends the resulting encrypted data to Bob. Bob uses the session key to decrypt the message, thereby recovering the original message and confirming that there was no tampering while the message was in transit.



Encrypted communication systems uses different cryptographic keys for different purposes. Each party has a long-term identity key, which is used in the initial handshake phase to authenticate the party's identity and negotiate an initial session key. If a malicious party learns Alice's long-term identity key, this would allow that party to impersonate Alice in the future, but it would not allow decryption or tampering with messages sent in a non-impersonated session.

Session keys are used to protect individual messages that flow between the two parties. If a malicious party learns a session key, the malicious party can decrypt or tamper with messages encrypted with that session key. It is common for systems to switch to new session keys frequently, to limit the damage that could result from loss of any particular session key. Many systems switch to a new session key for every message, so that loss of a session key compromises only a single message. Once a new session key has been chosen, all copies of the old session key are erased.

State-of-the-art encrypted communication systems are designed to be resilient, in the sense that there is no single secret key which, if compromised, would allow access to all messages. The consequences of a malicious party getting access to a secret key will depend on which key is compromised. A malicious party who somehow gets access to a party's long-term identity key will be able to impersonate that party in the future but will not be able to decrypt old messages. A malicious party who somehow gets access to a session key will be able to decrypt all messages that were encrypted with that session key, but will not be able to decrypt messages sent with older or newer session keys, and will not be able to impersonate anyone. The use of multiple keys, and the practice of switching session keys frequently, limits the harm that results from any one key being compromised.

Discussion: Common Threads in Modern Encryption Practices

Several common threads run through the design of modern encryption applications:

- Encryption does more than just keeping secrets. In addition to protecting the confidentiality of data, it protects integrity, allowing detection of any unauthorized attempt to tamper with the data. In some applications, integrity is the main reason for using encryption.
- Encrypted communication does more than just protect the message itself from decryption or tampering. It also allows the two parties who are communicating to authenticate each other's identities. In some applications, authentication of identity and of the source of messages is the main reason for using encryption.
- Information is often protected by erasing all copies of the key that was used to encrypt it. In encrypted storage, the storage key and Alice's passcode are erased when the device is locked, thereby ensuring that a malicious party who gets full access to the device while the device is locked cannot decrypt the data. In encrypted communication, the session key used to encrypt a message is erased as soon as the message has been decrypted by the recipient, thereby ensuring that a malicious party who recorded the encrypted data has no hope of getting the key that would enable decrypting it.