MITRE eCTF 2025 Design Document

Team ID MichState
School Michigan State University

Members

Udbhav Saxena
Maksim Savich
Samuel Lain Hedden
Fatima Saad
Matteo Krivitzky
Felipe Marques Allevato
Raj Dhullipalla
Aashish Harishchandre
Caroline Huang
Adrian Self
Charles Selipsky
Doruk Alp Mutlu
Mariana Vangelov
Jonathan Bolton
Shamar Dotson
Aditya Chaudhari
Lucas Bosca
Conner OSullivan

Advisor Dr. Qiben Yan

Overview of our System

The goal of our system is to secure a satellite TV stream that is broadcast to a set of subscribed decoders, ensuring decoders are only able to decode frames with timestamps that lie within the range permitted by their subscription. We achieve this by using a **Frame Key Derivation Tree**, a binary tree of height 64 where each of the 2^{64} leaf nodes represents a symmetric password used for encoding a frame with a corresponding 64-bit timestamp.

Each channel has its own tree generated by using a random root node value, and hashing with unique markers for deriving the left and right children. For a given range of frames allowed in a subscription, a set of ancestor nodes in the tree is selected which derive all corresponding leaf nodes, and is shared in the form of a subscription package. This makes it infeasible for a decoder to deduce the encryption key for a frame outside its given range.

We additionally use public-key cryptography with a host secret key and a corresponding public key distributed to all decoders. This is used to sign and verify the integrity of the subscription packages and frames, protecting against forgery by unverified third parties.

Build Process

Build Environment

The standard Python virtual environment from the reference design is used, with the addition of the PyCryptodome package for cryptography functions. For a single deployment, the following keys will be generated and stored in the Global Secrets:

- A host secret and public key pair H_{SK}, H_{PK}
- A decoder derivation key D_{DK} used for deriving symmetric keys per decoder via HKDF.
- Channel Root Keys $C_{n,root}$ where n is the channel ID, which are used to derive children nodes for their key derivation trees.

Build Decoder

When building the decoder, it is provided with the following secrets:

- A decoder key $D_{decoder}$
- The host public key H_{PK}
- The root key for the emergency channel $C_{0,root}$

Key Management In-Depth

Our system uses an asymmetric signing key pair, a unique secret corresponding to each decoder, and a unique secret for every (channel, timestamp) combination: the unique frame key.

Because the cardinality of the set is $1 + 2^{32} + 2^{32} \times 2^{64} \approx 2^{96}$, we cannot pre-generate and store this many keys during secret generation time; instead, we designed a key derivation strategy.

At secret generation time, the list of channels which are to be supported are already given. Therefore, we generate a secret key corresponding to each channel $C_{n,root}$ at this time and store a mapping from channel id to channel secret.

Unlike the channel list, the set of decoder IDs to be supported is not known at secret generation time. Therefore, we generate a single root decoder key D_{DK} at secret generation time, and use this root key along with the decoder's unique id d to derive a unique decoder key D_d at decoder build time.

For integrity verification, an asymmetric signing keypair $\{H_{SK}, H_{PK}\}$ is generated at secret generation time, and the public key is distributed to each decoder, while the private key is held by the encoder alone.

Frame Key Derivation Tree

The frame key was the most challenging to implement; we need to have a unique frame key for every possible (channel, timestamp) combination, and each decoder authorized to listen to a given channel must receive all the frame keys for that channel from the start timestamp to the ending timestamp through its subscription package.

In order to accomplish this, we create a tree structure for key derivation. The channel secret $C_{n,root}$ serves as the root of the tree, and there are 2^{64} leaf nodes which are the frame keys of that channel for each timestamp. This tree is again too large to store, but any node can be derived random-access from any ancestor node. Each child node in this binary tree is the hash of the parent node plus a direction, left or right. This means that no parent may be derived from a child (for a preimage-resistant hash), but the descendants may all be derived from the ancestor. Note that this is not a Merkle tree, where the parent is derived from the children, but the opposite, where the children are derived from the parent.

In our subscription package, we convey the appropriate covering of the binary tree from the leftmost leaf node to the rightmost leaf node required by the start and end timestamps. This covering contains no parent of a disallowed node, but otherwise contains as few nodes as possible by giving nodes as high in the tree as possible. The height of this tree is 64, and typical coverings do not exceed 80 nodes. The following diagram illustrates the tree structure:

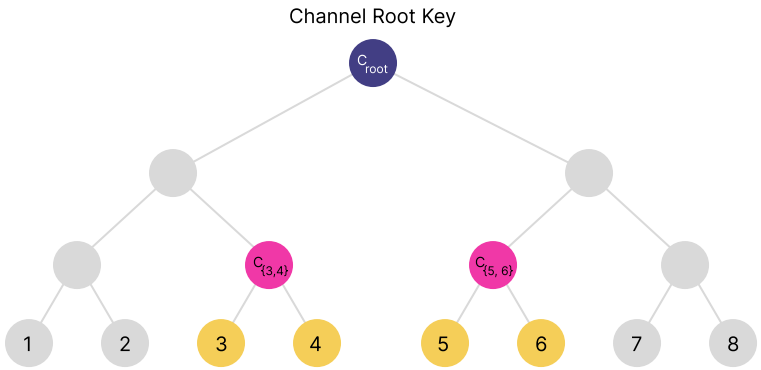

Figure 1: A channel key derivation tree for a channel with 8 frames)

Figure 1 shows a simplified scenario where a channel has 2^3 frames, or 3-bit timestamps. In this case, a subscription for the range of frames 3-6 (inclusive) can be shared by sharing the highlighted nodes $C_{\{3,4\}}$ and $C_{\{5,6\}}$. The maximum number of nodes necessary to share for an arbitrary tree is proportional to the height of the tree, which greatly reduces the amount of secret keys to be shared for a channel with 64-bit timestamps (2^{64} frames).

Algorithms

We use the following algorithms in our system:

- ED25519 for asymmetric signature and verification
- HKDF-SHA512 for decoder key derivation
- MD5 Reverse Hash Tree for frame key derivation
- ChaCha20 for encryption and decryption

Security Requirements

Security Requirement 1

An attacker should not be able to decode TV frames without a Decoder that has a valid, active subscription to that channel.

Every frame for a channel c with timestamp n encoded by the Encoder uses a unique key $C_{c,n}$, which can only be derived with an appropriate ancestor node present in a valid subscription package for a frame. Furthermore, subscription packages for each decoder d are encrypted using the decoder-specific key D_d derived from D_{DK}. This means a subscription for a different decoder leaks no information about what is needed to decode the channel.

Security Requirement 2

The Decoder should only decode valid TV frames generated by the Satellite System the Decoder was provisioned for.

Every frame in the broadcast is signed using an asymmetric key, and is verified by the decoder with knowledge of the host's valid public key. The host secret key H_{SK} is not shared outside the deployment, and it is infeasible to forge a signature in a frame or subscription package without it.

Security Requirement 3

The Decoder should only decode frames with strictly monotonically increasing timestamps.

The firmware will keep track of the last frame received for each channel in volatile memory. If a received frame has a timestamp lower than this value, the decoder will reject decoding the frame.