A Cryptanalysis of the High-bandwidth Digital Content Protection System

PRESENTED AT ACM-CCS8 DRM WORKSHOP 11/5/2001

Scott Crosby
Carnegie Mellon University

Ian Goldberg
Zero Knowledge Systems

Robert Johnson   Dawn Song   David Wagner
University of California at Berkeley

November 9, 2001

What is HDCP?

High bandwidth Digital Content Protection (HDCP) is a system for preventing access to plaintext video data sent over Digital Visual Interface (DVI). Any technique that allows access to the plaintext data is considered breaking the system.

I show that with the public and private keys from 40 devices and $O(40^2)$ work I can violate the design requirement—I can access the plaintext. Furthermore, with the 40 sets of keys and at most $O(2^{40})$ offline work I can usurp the central authority completely.

The authentication process used in HDCP

The private key consists of a vector of 40 56-bit numbers.
The public key consists of a vector of 40 bits, half zero, half one.
Each device A, B, B^i has these keys assigned by a central authority.
Dot products are computed in the ring $\mathbb{Z}/2^{56}\mathbb{Z}$.

$$
\begin{aligned}
A \to B: &\quad \vec{A}_{public}, n_A \\
B: &\quad K_{shared} = \vec{A}_{public} \cdot \vec{B}_{private},\ R = h(K_{shared}, n_A) \\
B \to A: &\quad \vec{B}_{public}, R \\
A: &\quad K_{shared'} = \vec{B}_{public} \cdot \vec{A}_{private},\ R' = h(K_{shared'}, n_A) \\
A: &\quad \text{Verifies } R = R' \text{ and } \vec{B}_{public} \text{ is not on a blacklist.}
\end{aligned}
$$

The shared secret K_{shared} is reused later in the protocol.

The flaw

HDCP uses a linear system for generating the shared secret.

$$\vec{A}_{public} \cdot \vec{B}_{private} = K_{shared} \stackrel{?}{=} K_{shared'} = \vec{B}_{public} \cdot \vec{A}_{private}$$

The flaw is that any device whose public key is a linear combination of public keys of other devices will, when assigned a private key that's a similar linear combination of the other devices private keys, successfully authenticate.

This flaw is fundamental, and cannot be worked around.

Proof of the break

I assume we have enough private keys $\vec{B}^i_{private}$ whose public keys span $M \subseteq (\mathbb{Z}/2^{56}\mathbb{Z})^{40}$, the module generated by all public keys assigned by the central authority. I assume that all of these devices will successfully authenticate with A.

As the subspace is 40 dimensional, a set of at most 40 keys will be enough.

Consider any device C with $C_{public} \in M$ whose public key and private key are any non-zero linear combination of B^k's public and private keys.

Say, $\vec{C}_{public} = \sum_{i=1}^{40} a_i \vec{B}^i_{public}$

And, $\vec{C}_{private} = \sum_{i=1}^{40} a_i \vec{B}^i_{private}$

Let A and C authenticate

$$K_{shared} = \vec{A}_{public} \cdot \vec{C}_{private} = \vec{A}_{public} \cdot \sum_{i=1}^{40} a_i \vec{B}^i_{private}$$

$$= \sum_{i=1}^{40} a_i (\vec{A}_{public} \cdot \vec{B}^i_{private}) = \sum_{i=1}^{40} a_i K^i_{shared}$$

$$K_{shared'} = \vec{C}_{public} \cdot \vec{A}_{private} = \vec{A}_{private} \cdot \sum_{i=1}^{40} a_i \vec{B}^i_{public}$$

$$= \sum_{i=1}^{40} a_i (\vec{A}_{private} \cdot \vec{B}^i_{public}) = \sum_{i=1}^{40} a_i K^i_{shared'}$$

When A and C authenticate

$$\sum_{i=1}^{40} a_i K^i_{shared} = K_{shared} \stackrel{?}{=} K_{shared'} = \sum_{i=1}^{40} a_i K^i_{shared'}$$

We know that $K^i_{shared} = K^i_{shared'}$ for all i because by assumption, B^i's successfully authenticate with A. Therefore, $K_{shared} = K_{shared'}$ and this authentication succeeds.

Thus, for any device C with in $C_{public} \in M$, we can decrypt any stream in $O(40^2)$ work by rewriting $\vec{C}_{public}$ as a linear combination of $\vec{B}^i_{public}$.

Furthermore, we can forge keys in at most $O(2^{40})$ work by enumerating all well-formed public keys and seeing if they're in M. This completely usurps the central authority; we can do all that they can do.

In practice, I expect we can usurp the authority without this work factor.

Attacks that don't need the ability to forge

These avoid the hassle of finding well-formed public keys in M.

We don't have to forge to be able to clone, eavesdrop, or avoid the blacklist. We do have a ready source of well-formed public keys: All devices come with such a key, and receivers supply their public key on demand.

We can masquerade as any receiver for whom we have their public key. Just sit in the middle between them and the transmitter, snarf the receivers public key, and use it as our own.

This can be done in $O(40^2)$ work; by rewriting $\vec{C}_{public}$ as a linear combination of $\vec{B}^i_{public}$.

What about modifications? Longer key vectors?

If there are longer key vectors, the attacks continue to work, but we may not be able to forge because of the pesky well-formedness condition on public keys.

For example, they may chose a submodule where it may be hard to find a new key $\vec{C}_{public} \in M$ that is well-formed. The best general algorithm I've found for finding well-formed keys is brute-force.

A variant of this problem is in is NP-complete (reduction from subset-sum).

But, longer key vectors do not make us immune from any of my non-forging attacks. I can decrypt, clone, and avoid any blacklist.

What about modifications? How about certificates?

Modify HDCP to add cryptographic certificates onto public keys so that each device includes a digital certificate on the device's public key signed by the central authority with some conventional algorithm (e.g., RSA). The receiver sends their public key and the certificate of the public key, authentication proceeds only if the certificate is verified.

Call this protocol HDCP+certs.

This scheme is susceptible to the attacks that do not require forging.

I conjecture based on the incomplete specification available that DTCP's Restricted Authentication Protocol may be a variant of HDCP+certs. If so, this raises serious questions about its design.

Conclusion

HDCP's linear key exchange is a fundamental weaknesses. We can:

- Eavesdrop on any data
- Clone any device with only their public key
- Avoid any blacklist on devices
- Create new device keyvectors.
- In aggregate, we can usurp the authority completely.

The weaknesses are not easy to repair. Two proposed modifications are broken and still susceptible in $O(n^2)$ work and n sets of keys to:

- Eavesdrop on any data
- Clone any device with only their public key
- Avoid any blacklist on devices