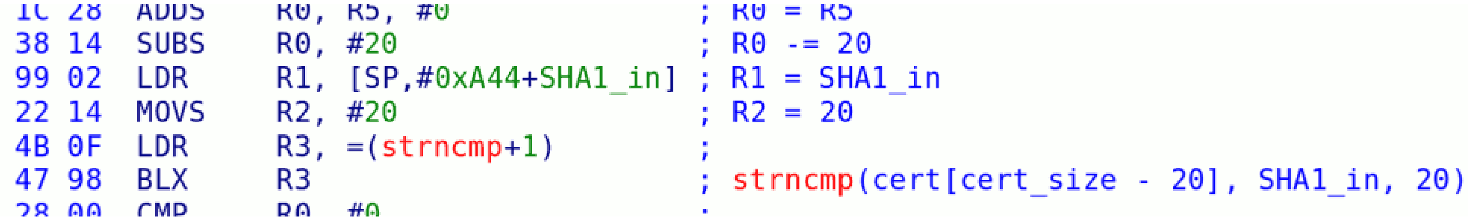
EXHIBIT J

DECLARATION OF RYAN BRICKER IN SUPPORT OF *EX PARTE* MOTION FOR TEMPORARY RESTRAINING ORDER AND ORDER TO SHOW CAUSE RE PRELIMINARY INUNCTION; ORDER OF IMPOUNDMENT

HackMii

Notes from inside your Wii

- front page
- archives
- about
- RSS



```
1C 28  ADDS    R0, R5, #0                ; R0 = R5
38 14  SUBS    R0, #20                   ; R0 -= 20
99 02  LDR     R1, [SP,#0xA44+SHA1_in]   ; R1 = SHA1_in
22 14  MOVS    R2, #20                   ; R2 = 20
4B 0F  LDR     R3, =(strncmp+1)          ;
47 98  BLX     R3                        ; strncmp(cert[cert_size - 20], SHA1_in, 20)
28 00  CMP     R0  #0                    ;
```

- ## Categories
  - dsi
  - Other consoles
  - Wii
- ## Archives
  - December 2010
  - November 2010
  - September 2010
  - August 2010
  - July 2010
  - June 2010
  - May 2010
  - April 2010
  - February 2010
  - January 2010
  - December 2009
  - November 2009
  - October 2009
  - September 2009
  - August 2009
  - July 2009
  - June 2009
  - May 2009
  - April 2009
  - March 2009
  - February 2009
  - January 2009
  - December 2008
  - November 2008
  - October 2008
  - September 2008
  - August 2008
  - July 2008
  - June 2008
  - May 2008
  - April 2008
- ## Admin
  - Register
  - Log in

← Thanks, Waninkoko. BrickMii? →

Keys, keys, keys.

April 15th, 2008 by bushing · 23 Comments

By popular request, here's an explanation of the different encryption keys that are used on the Wii.

AES Keys: The Wii uses 128-bit (16-byte) symmetric AES (aka AES-128-CBC) for most encryption.

- Common key (ebe42a225e8593e448d9c5457381aaf7): This is the "shared secret" that we extracted with the Tweezer Hack. This key is known by all Wiis, but is never used, directly, to encrypt anything. Instead, all titles are encrypted with a random AES key; this key is then encrypted with the Common key and then stored inside a ticket. The ticket is then transmitted along with the content — on discs, it's part of the "certificates" found before the encrypted data starts. Thus, knowing the common key allows you to decrypt most Wii content, as long as you have the right ticket. This key is stored in the OTP area inside the Starlet ARM core inside the Hollywood package.
- SD key (ab01b9d8e1622b08afbad84dbfc2a55d): This is another shared secret — also stored on the Hollywood, but also found plenty of other places, including inside the firmware images. This key is used by the System Menu (1-2) to encrypt anything before writing it out to the SD card, and it's used by 1-2 to decrypt anything read from the SD card. This is done mainly for the purpose of obfuscation, to keep people from examining savegames. It's worth noting that all Wii games save their data to the internal NAND — no game supports loading or saving data directly to SD. This frees game writers from the requirement of handling this step themselves; they just write the savegame data, unencrypted and unsigned, to their title-data directory inside the NAND filesystem; the system menu then handles everything else. (The real reason for this is probably that it allowed Nintendo to make a system where they didn't have to expose the details of this encryption — or any encryption — to their licensed game developers.) This key is also stored in OTP, and in several places in IOS (for no apparent reason). If you're using Segher's tools, you may also be interested in the SD IV (216712e6aa1f689f95c5a22324dc6a98) and the MD5 blanker (0e65378199be4517ab06ec22451a5793), both of which are stored inside the 1-2 binary.
- NAND key (varies): This AES key is used to encrypt the filesystem data on the actual NAND chip itself; it is probably randomly generated during manufacturing and is also stored in the OTP area of the Starlet. This key is used to prevent the contents of the NAND filesystem from being read using a flash chip reader. Nintendo may or may not actually record this key anywhere, since they (theoretically) don't need to ever use it. In fact, in some similar systems, keys like this are generated automatically by the device itself and (theoretically) never leave it — the Wii shares some design prinicples with HSMs, but it certainly doesn't manage to be one. This is another OTP key.

RSA keys: The Wii uses RSA-based authentication in several different places. This is fundamentally different than the AES encryption used for data-hiding, because RSA is an asymmetric cipher, meaning there are no shared secrets — nothing to be extracted from the Wii. The only RSA keys stored on the Wii are public keys, used to verify authenticity of content.

- CP: Content Protection? This key is used to sign the TMD associated with every title. The TMD contains a SHA1 hash of the contents of that title, proving that it had not been modified. My 24c3 presentation was done by injecting a new .DOL into a Lego Star Wars disc and then forging the signature on its TMD, using a flaw originally discovered by Segher. After that presentation, people eventually discovered the common key needed to decrypt update partitions, allowing others to analyze / disassemble IOS. xt5 (who I had the pleasure of meeting at 24c3) was then able to find the same flaw and implemented it in his Trucha Signer. In fact, from disassembling his code, the core part of it was almost identical to our never-released code — great minds think alike, eh?
- XS: ”Access”? This is the key that signs tickets, which contain the title keys for individual titles.
- CA: Certification Authority: This key signs both the XS and CP keys.
- MS: "Master?" This key is used to sign the certificate that contains a copy of your Wii's public ECC key. This certificate is then appended to savegames on SD cards, so that any other Wii can verify that the key was issued by Nintendo.
- Root: This is the "grand master key", which signs the CA key. The public half of this can be found here.

ECC keys: The Wii uses Elliptic Curve Cryptography in a few select places — primarily, it uses this when it signs savegames before writing them to SD card. ECC is used in ways similar to RSA, but it's somewhat newer and much faster to run on an embedded system.

Other: For lack of a better place to put it, there is also an HMAC key — a 20-byte value that is used in a SHA1-based HMAC of the NAND flash contents to prevent them from being tampered with. This is a commonly used scheme in embedded systems, where a device wants to "sign" something itself, for itself. There are no public vs private keys here — you need to know this value in order to verify the hash, and you need the same value to generate the hash. This isn't appropriate for communications between two people, but is perfectly fine for letting the Wii test to see if the chip was pulled, rewritten, and resoldered.

Key storage: The public keys are stored in various places — these aren't sensitive, so they don't really need to be concealed (although at least one of them needs to be protected from modification, and it can then sign the others). The rest are stored in two places:

- Hollywood SEEPROM: After meeting him at 24c3, bunnie was kind enough to decap some chips for me, including a Hollywood. One of those chips is 2kbit serial EEPROM, which stores the MS signature on the the ECC key.
- One-Time Programmable Area: Inside the Starlet ARM core, there are a bunch of things:

1. SHA1 hash of boot1
2. Common key
3. ECC private key
4. NAND HMAC
5. NAND AES key
6. RNG seed
7. other stuff we can't yet decipher

All of that info comes from tmbinc, who recovered it with a method he described here.

Tags: Wii

23 responses so far ↓

- 1 **Phredreeke** // Apr 16, 2008 at 4:41 am

  Interesting read 😃 Keep up the good work

- 2 **Nobody** // Apr 16, 2008 at 6:24 pm

  Bushing,

  Can you tie some of the official key names with those used in Trucha Signer?

  "boot1 key" = NAND key?
  "common key" = key.bin
  "sd key" = SD Key
  "sd iv" = ?
  "md5 blanker" = ?

  Excellent blog, by the way. It's become one of my top reads.

- 3 **marcan** // Apr 16, 2008 at 7:31 pm

  SD IV and MD5 Blanker are just arbitrary 16-byte constants stored inside the system menu. They aren't "keys" per se and they aren't sensitive.

  The boot1 key is separate from the NAND key. Common key is indeed key.bin.

- 4 **Dasda** // Apr 17, 2008 at 12:29 pm

  The IV is just an XOR on the first 16 bytes of the decrypted file, to make them as they were before the encryption (I really don't know why, I don't think this is something related to security though).

- 5 **Odb718** // Apr 19, 2008 at 10:24 am

  Holy mother of keys! I had NO idea there were that many. This is worse then a janitor at a public school. I thought there'd be maybe 5 keys used.

- 6 **TD-Linux** // Apr 19, 2008 at 12:20 pm

  Wow, a very nice explanation of the mess of a security system the Wii has! I especially liked your separation of the different kinds of keys (symmetric, asymmetric, etc). I'll go see if this is on the wiki somewhere. If it isn't, it's very excellent information, and someone (possibly me) should port it over into mediawiki syntax.

- 7 **Newbie** // Apr 19, 2008 at 3:57 pm

  Bushing,
  Can you clarify, what keys are unique for each console, please?

  Another (or same?) question,
  I tried to write a "Wii save games editor", basically nothing more than a front end to Segher tools with some compare/edit capabilities.
  Based on your post, the only part I missed was ECC keys, right?

Now the question is: If we ever get those keys, what else could they be used for, (or if you prefer MISused for)? I hope they are not used in any way in Virtual Console…

- 8 **Dasda** // Apr 19, 2008 at 4:12 pm

  Newbie: ECC keys are used for saves and virtual console games when you put them on SD-cards.

- 9 **Newbie** // Apr 20, 2008 at 7:50 am

  Dasda,
  Do you mean, the process will be the same to saves and VC games (VCG)?
  Copy to SD – [optionally publish on internet] – unpack – [optionally patch] – pack using your keys – copy to your Wii, right?
  I thought, VCG have more "layers of protection" than saves…
  Since there is no problem with "unpacking" saves, why pirates bother with NAND FS dumper, if the "unpack" process is the same for VCG and saves?

- 10 **chris** // Apr 20, 2008 at 2:15 pm

  very interesting indeed, it makes me want to learn about cryptography…. however, I'm not sure to understand exactly the difference between a ticket and a tmd file ? and what is exactly a wad ? mzybe the next lesson could be about the different wii filetypes 😉

- 11 **Dasda** // Apr 22, 2008 at 11:01 am

  Newbie: These keys are only used to sign channels you copy from your Wii, not the channels included in games or updates. The ones you copy from your Wii still contains the same signatures as the channels (you can only sign to verify that the VC was from your Wii, not that it is authentic).

  Chris: A wad file basically is a TMD, ticket, the content files and the certificate chain stored after each other (that's not the correct order). It contains a small header holding the sizes of each part, and they are rounded up to 64 bytes I think.
  The ticket contains the key used to decrypt and the TMD contains information about the content files.

- 12 **chris** // Apr 27, 2008 at 10:30 am

  @Dasda: thanks for the precisions

  I have another question though: I've come across a tool called WAD uninstaller that seems to uninstall a previously installed channel

  Even if I think I understand how the install process is going on (the WAD content is probably based on a title content + a title installer ?), i don't understand why a WAD uninstaller is needed… is that not enough to use the channel delete function in the Wii menu ? DOes this mean the WAD is also adding some stuffs behind the title ?

- 13 **bushing** // Apr 28, 2008 at 1:14 am

  @Dasda: IV: http://en.wikipedia.org/wiki/Initialization_vector

  @Newbie: There are three keys that are unique to each Wii: NAND AES, NAND HMAC, ECC.

  VC games do have an extra layer of protection, in that the ticket that belongs to each game is keyed specifically (and individually) for each Wii.

  @chris: A wad uninstaller is needed because not all channels can be deleted — I think there's either a range of title IDs, or a bit that can be set in the TMD, that indicates that a channel is a "system channel" which cannot be deleted. This is why people can't fix the "duplicate channel problem" themselves.

  That aside, when you do delete something, it still leaves some bits around — at least the ticket, and possibly the TMD and the directory structure, too. None of this should be a problem — but hey, you never know…

- 14 **Newbie** // Apr 28, 2008 at 6:41 pm

  @bushing
  Thanks a lot. In my post (#7), it was another question as well, could you answer it please?
  ———
  Now the question is: If we ever get those keys, what else they could be used for, (or if you prefer MISused for)

- 15 **bushing** // May 6, 2008 at 1:32 pm

  @Newbie: I didn't answer your question, because Dasda did, in #8. To rephrase his answer:

  ECC keys are only used in one place: by the system menu, when copying things to or from the SD card. So, when you copy a savegame or a VC game from the NAND to an SD card, it gets signed with your console's ECC key.

  When you copy a savegame or a VC game from the SD card to the NAND, the System Menu checks to make sure the ECC signature is valid — among other things. The main check is "do I have a TMD for this game" (for savegames) or "do I have a valid ticket for this game" (for VC games). It also does the usual RSA signature check, etc.

So, no, there's not much you can do with the ECC key except for hack savegames, and there are easier ways to accomplish that. Namely, edit the contents of the file on NAND and then use the menu to copy it to SD card for you, or there's an ES call (ES_Sign) which will just do this directly from any program you write.

- 16 **Newbie** // May 7, 2008 at 8:09 pm

  @bushing & dasda,
  Thanks for your answers!

  I didn't answer your question, because Dasda did, in #8.
  I guess I'm not smart enough to get it from Dasda reply!

  The main check is "do I have a TMD for this game" (for savegames)
  Does is mean we cannot copy a savegame [to Wii] until the game DVD is inserted at least once?

  So, no, there's not much you can do with the ECC key except for hack savegames
  Good to know. I thought nobody want to share the way to extract them because of piracy reasons. Apparently (from your later posts!) it's just not quite simple thing to do!

  and there are easier ways to accomplish that. Namely, edit the contents of the file on NAND and then use the menu to copy it to SD card for you, or there's an ES call (ES_Sign) which will just do this directly from any program you write.
  As of now it seemed a much easy way to "copy to PC – decrypt – edit – encrypt – copy to Wii", than write full blown save game editor as native Wii program (at least for me!). But there is no doubt, in the very near further we will see that program.

- 17 **thiefstar** // Jan 13, 2010 at 2:05 am

  @bushing

  Hi,

  Some people said:

  opne the brickednand.bin and

  Found "if mdck =*************** then"
  Sentence, in which "***************" is your CPUKEY.
  and Open the NAND.BIN with TrueCrypt v6.3a For Linux
  Found "if [email = mdck = @ then] mdck =########## then [/ email]"
  Replace ########## with the *************** and save
  finally flash the NAND.BIN to nand flash and soldering the nand flash. it will done.

  It is possible?

- 18 **bushing** // Jan 16, 2010 at 5:48 am

  @thiefstar: This came from http://www.91wii.com/thread-26561-1-1.html, right?

  It's bullshit. It makes no sense, the guy is making it all up. I'd try to explain why, but there's not even anything there to disprove — it does not even slightly resemble reality. Why do people write up posts like that?

- 19 **thiefstar** // Jan 18, 2010 at 7:56 pm

  @bushing

  yes. this came from 91wii.com
  Author mentioned in the text. He said that did not write all the steps.

  This is very interesting

- 20 **thiefstar** // Jan 18, 2010 at 10:01 pm

  I have Contacted that the author. Full brick repair, he said only 50% success rate. But 003 is 100%(Of course. He would not tell me repair methods. Because he was the store staff)
  it seems that he does not replace the nand file. I think he should be to amend the IOS70 data of nand file.

- 21 **feraligatr** // Jan 26, 2010 at 7:20 pm

As far as I can tell, he is nothing other than a fraud. His words are totally non-sense.

- 22 **KingLewy** // Feb 1, 2010 at 1:54 am

  I thought the only way to repair a bricked Wii was to already have a dump of your Wii's NAND before it bricked, right?

  That says to me that anybody claiming they can repair your bricked Wii is lying. They'd need you to have already backed up your Wii's memory (for which not everyone is as insightful) and if you can do that, then you can repair it yourself, yeah?

- 23 **DCX2** // Sep 27, 2010 at 7:02 pm

  So there's a pretty severe Metroid: Other M glitch that can kill your game. I would like to make a PC application that can take MOM save games and fix the glitch. Preferably, this would work without any homebrew at all, so that Joe Sixpack could fix his MOM glitch without HBC. However, I'm having some trouble digesting the key info…I'm not really a crypto person.

  Is it possible to do this with just the SD key? Do I need the ECC key? Can I fake the ECC key somehow?

You must log in to post a comment.

- ## Search It!

  Search

- ## Recent Entries

  - Open-source USB Analyzer / 27C312.19
  - Developers, Developers, Developers!11.6
  - Insert Startup Disc9.22
  - The scope of Homebrew Channel8.19
  - The USB2 Release8.14
  - Theming the Homebrew Channel8.11
  - HackMii Installer v0.77.26
  - System Menu 4.3 update6.24
  - of homebrew and "trusted computing" / antipiracy5.15
  - "Pandora's Xbox: The changing community of the modern console"5.1

- ## Blogroll

  - iPhone Dev Team Blog
  - mist's blog
  - root labs
  - tmbinc's blog
  - Wiibrew wiki
  - xorloser's blog

close

Galleries
of
by