Fundamentos do sistema de encriptação usado pelo hardware da Xbox

por

Fernando Vítor Pereira de Melo Tinoco
Filipe Gonçalves Barreto de Oliveira Castilho
Departamento de Engenharia Informática
Universidade de Coimbra
3030 Coimbra, Portugal
ftinoco@student.dei.uc.pt
fgonc@student.dei.uc.pt

Resumo: *Este artigo descreve os fundamentos do sistema de encriptação usado pelo hardware da Xbox. Um boot block escondido dentro do sistema ASIC é responsável pela descodificação e verificação de vários pedaços de uma memória de leitura (mas na qual é possível fazer uma actualização através de um flash dessa memória). Este boot block é disfarçado pela presença de um outro boot block falso na memória de leitura (ROM) externa. O código contido dentro do boot block escondido é transferido para o processador através de um conjunto de busses de grande velocidade, onde é descodificado usando hardware específico para esta funcionalidade. Este artigo termina com algumas recomendações para melhorar o sistema de segurança da Xbox. Ao longo deste artigo o leitor irá perceber que o uso de um bus de grande desempenho não é uma medida de segurança suficiente.*

Palavras-Chave: *Xbox, bootloader, Segurança na Xbox, flash rom, boot block, HyperTransport bus.*

1. Introdução

Todos os sistemas de encriptação são baseados num segredo, como por exemplo, uma chave. Independentemente da codificação usada, a segurança do sistema de encriptação só é segura consoante o nível secretismo da chave usada. Consequentemente, alguns dos ataques mais eficazes a estes sistemas de encriptação não envolvem análise da codificação usada, mas sim a procura de falhas no sistema que é responsável pelo armazenamento das chaves. Os sistemas de encriptação baseados numa codificação simétrica são particularmente vulneráveis a ataques de protocolo, uma vez que, o transmissor e o receptor têm de ter os dois, uma cópia da mesma chave de segurança. Apesar da dificuldade do manuseamento de chaves nestes sistemas simétricos, estas mantêm-se atractivas devido à simplicidade dos seus algoritmos e ao grande *throughput* quando comparadas a sistemas públicos de codificação de chaves.

O manuseamento de chaves baseadas no sistema de codificação simétrico torna-se especialmente problemático quando o lado do receptor não é de confiança ou quando está numa posição onde a sua segurança pode ser facilmente comprometida. É nestas alturas que se usa um hardware específico para evitar modificações no sistema de segurança. Existem muitos sistemas de segurança a aplicar esta técnica mas com algumas variações, tais como a fechadura micro mecânica de 24 bits "Stronglink" da Sandia National Labs, o Clipper chip, o 4758 PCI Cryptographic Coprocessor da IBM, Smartcards criptográficos, Automatic Teller Machines (ATMs), e agora, as consolas de videojogos. No entanto, a confiança em sistema físicos inadequados para proteger segredos importantes é arriscado, e é um exemplo de como alguns dos sistemas já mencionados podem ser derrotados com métodos surpreendentemente simples e directos.

No caso da Xbox, os segredos que são protegidos são uma chave e um algoritmo para descodificação e verificação do *bootloader*. De seguida, o *bootloader* descodifica e verifica a imagem do *kernel*. O *bootloader* e a imagem do *kernel* estão contidos numa *FLASH ROM*. O *kernel* de seguida verifica a autenticidade e integridade das aplicações que executa. Consequentemente, uma cadeia de confiança é criada, de baixo para cima, a partir de uma "semente" de confiança. Esta "semente" – a chave secreta e o algoritmo – é plantada fisicamente num *boot block* secreto e seguro.

A arquitectura da Xbox resulta na aplicação de uma grande quantidade de dispositivos idênticos, que são constituídos pela mesma informação secreta. Na secção seguinte é ilustrado como a segurança de um sistemas destes pode ser comprometida facilmente, mesmo que os seus segredos sejam protegidos por um hardware específico que protege contra modificações alheias ao sistema e também escondido por algoritmos de elevada complexidade.

2. Sistema de Encriptação da Xbox

O protocolo de encriptação da Xbox apresenta uma grande defesa perante as modificações pouco seguras da *FLASH ROM***.** A Xbox arranca a partir de um *boot block* escondido de 512 bytes que se encontra integrado no sistema *southbridge ASIC* (o "*MCPX*") (ver Figura 1). Este *boot block* desempenha as seguintes funções, realizadas pela seguinte ordem:

- Inicialização do chipset da consola;
- Liga as caches do processador;
- Desencripta o *bootloader* do *kernel*, contido na *FLASH ROM;*
- Confirma se a desencriptação foi bem sucedida;
- Passa ao *bootloader* do *kernel*, entretanto já desencriptado.

Após estas acções, o *bootloader* executa mais algumas operações de inicialização do sistema: desencripta uma imagem do *kernel* a partir da *FLASH ROM*, descomprime e verifica a imagem desencriptada, e em seguida entra no *kernel*. A chave de desencriptação do *kernel* é guardada dentro da imagem do *bootloader*. De referir que o *boot block* escondido está estruturado para que a chave de decifração do *bootloader* nunca seja escrita para a memória principal, evitando assim um eventual ataque que envolva *eavesdropping* ao bus da memória principal. O *bootloader* é encriptado com RC-4 usando uma chave de 128 bit. O algoritmo de desencriptação e a chave são guardados no *boot block* escondido e executados pelo processador *Pentium*. Os *busses* entre o *boot block* e o processador não se

encontram encriptados, pois assume-se que são seguros devido à sua alta velocidade. A desencriptação da imagem do *bootloader* é verificada através da procura de um número mágico de 32 bit que se encontra perto do fim da *stream.* Esta verificação apenas assegura que a *stream* da codificação não se encontra corrompida. Assim, alguém que conheça a chave e o número mágico pode facilmente criar imagens do *bootloader* originais. É fácil de deduzir através da estrutura do código do *boot block* escondido que esta verificação simplista e pouco fiável foi utilizada porque não existia espaço para algo mais complexo e seguro. A verificação do número mágico pode também provocar problemas para criar um *bootloader* original que seja baseado numa chave obtida sem o conhecimento por inteiro do conteúdo do *boot block* escondido, como acontece pelo método de força bruta. No entanto, recorrendo ao método de força bruta para recuperar o *bootloader* estará provavelmente fora de questão. Dado este protocolo de arranque ser seguro, modificar somente o conteúdo da *FLASH ROM* dificilmente revelará alguma coisa de útil acerca da consola. Além disso, a *FLASH ROM* contêm um *boot block* falso que contêm algum código de inicialização. Aplicando o algoritmo do *boot block* falso nos conteúdos da *ROM* produz-se apenas "ruído".

3. Ultrapassar a Segurança Física

Nesta secção mostra-se de uma maneira cronológica o processo de *reverse enginering* do sistema de segurança físico.

Efectuar a leitura do conteúdo da *FLASH ROM* e fazer um *trace* da execução do processador começando no *boot vector*, mostrou ser um processo inútil, pois o conteúdo do *boot block* do *FLASH ROM* era um"isco", desenhado para evitar tal actividade. O código do *boot block* do *FLASH ROM* teve o mesmo seguimento que o código dentro do *boot block* secreto, mas o algoritmo de descodificação, as chaves e o início da localização do *ciphertext* estavam errados. Isto no início resultou numa grande confusão, mas mais tarde foi explicado pela descoberta de uma camada sobreposta com o *boot block*.

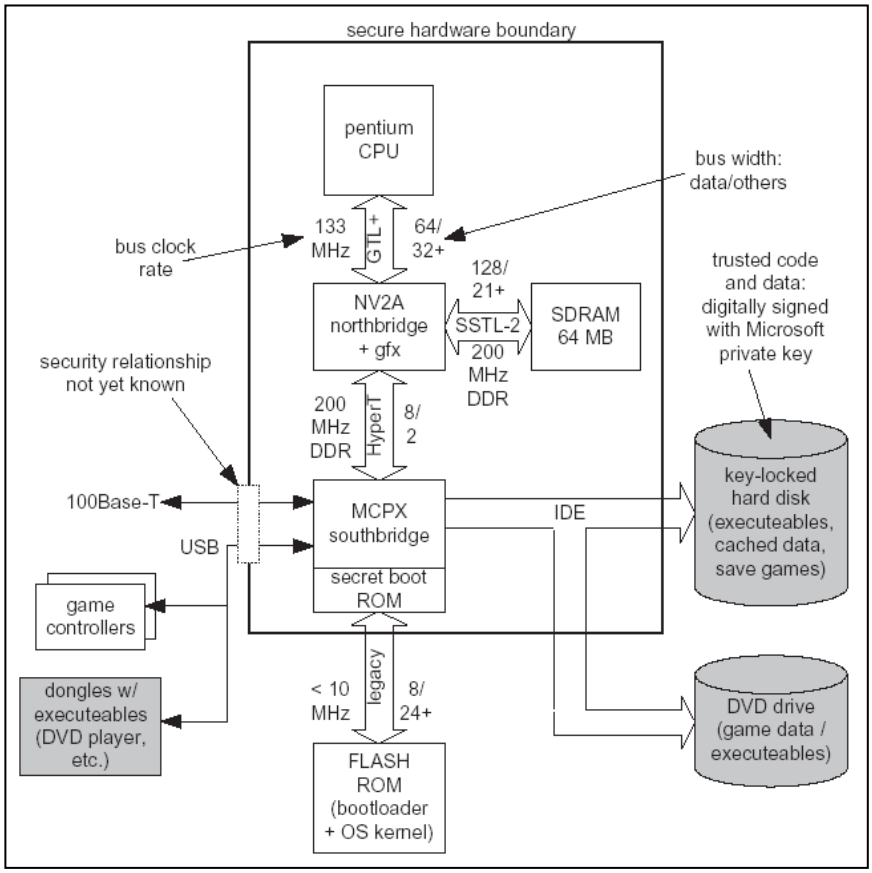

Figura 1 - Microsoft Xbox Hardware

A percepção da existência de um *boot block* secreto aconteceu aquando da observação do resultado da reescrita do vector *reset* do processador na *FLASH ROM*, que não tinha qualquer efeito na sequência de *boot* da Xbox. Isto levou a uma série de experiências que levou à descoberta da existência do tamanho do *boot block* secreto. Este bloco acredita-se que tenha um tamanho de 512 bytes, e esteja situado numa alta localização dentro da memória física do processador.

Os seguintes métodos foram considerados para extrair o conteúdo do *boot block* secreto:

- Desbloquear o *MCPX southbridge ASIC*;
- Analisando o bus da memoria principal para verificar se existem porções do *boot block* escritos na memória;
- Analisando o bus do processador *northbridge* usando um analisador lógico;
- Analisando o bus do *HyperTransport northbridge-southbridge* usando hardware especifico.

A aproximação directa do primeiro método foi rejeitada porque este *ASIC* aparenta ser fabricado num processo de 0.13µ com 6 ou 7 camadas metálicas (figura 2). A sondagem pela SDRAM também foi rejeitada baseada no facto de que muitos pinos iriam ter que ser verificados simultaneamente (128 pins). Por razões semelhantes, a sondagem do bus do processador *northbridge*, também foi rejeitado, pois seria necessário verificar 64 pins de dados.

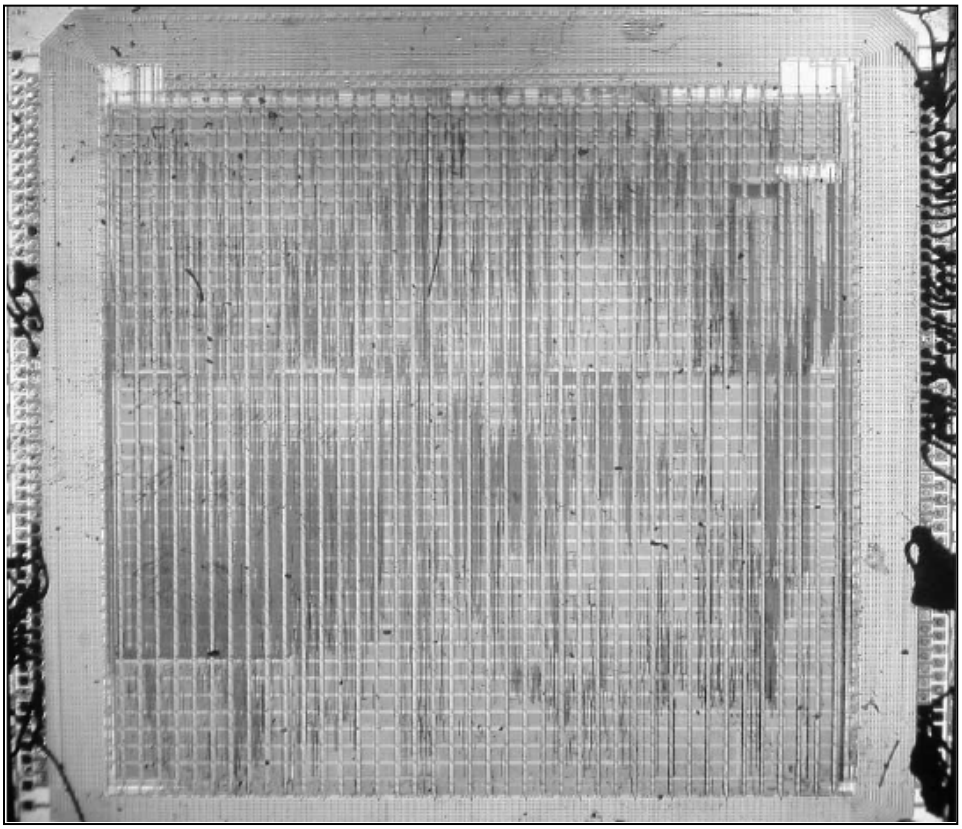

Figura 2 - MCPX Southbridge ASIC

O bus *northbridge-southbridge*, mostrou ser promissor devido à sua simplicidade. O bus tem baixo sinal (10 originais) e todos os *traces* do sinal são dispostos na motherboard da consola de uma maneira directa (*flow-through*). Os sinais do relógio para a transmissão e recepção estão bem identificados na motherboard. Os dados do *chipset* nVidia nForce são relativamente parecidos com os do *chipset* da Xbox, que indicam que o bus usa o protocolo *HyperTransport* (que antes era chamado de *Lightning Data Transport (LDT)*). A especificação deste protocolo é aberta e pode ser consultada on-line.

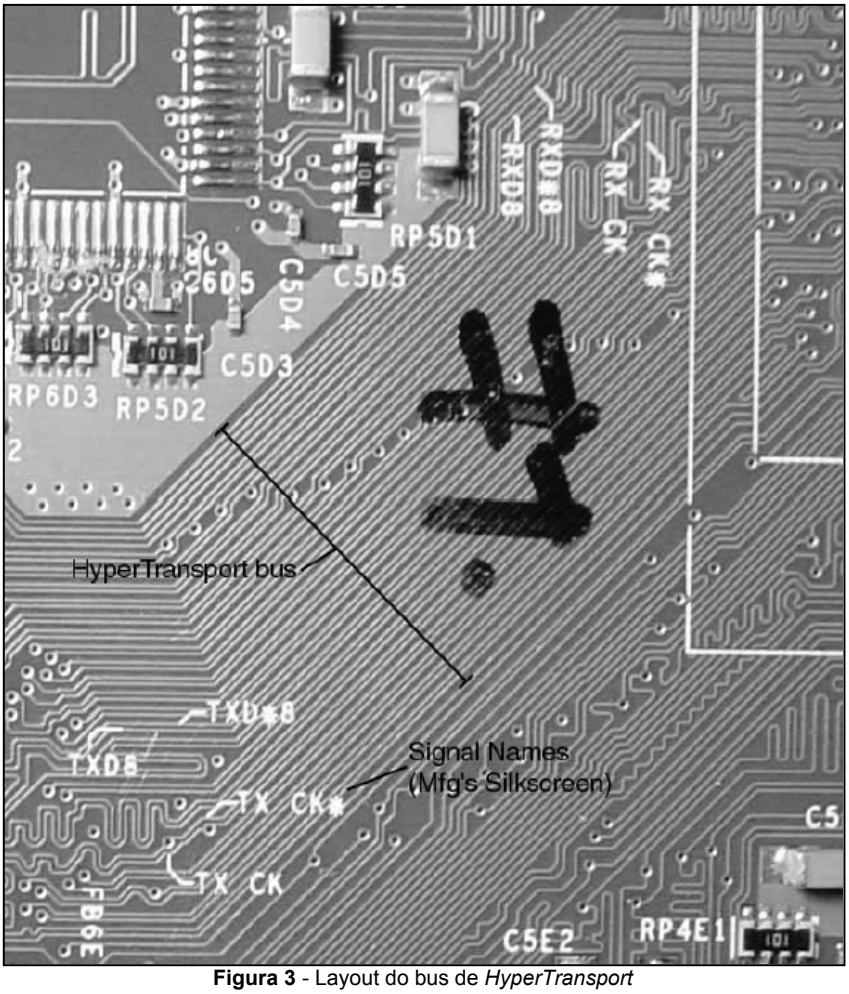

Figura 3 - Layout do bus de *HyperTransport*

A primeira dificuldade encontrada para analisar o conteúdo do bus de *HyperTransport* foi a sua elevada velocidade (200 MHz DDR) e o uso de sinais diferenciados (pois, existem poucos *scanners* que vêm preparados para analisar sinais diferenciados).

A solução encontrada para analisar o conteúdo do bus de *HyperTransport* do *northbridge-southbridge* foi construir um circuito especificamente para esse fim e ligá-la ao output da board de um FPGA.

Foi usado um conversor da Texas Instruments, o SN65LVDS386 LVDS-to-TTL, para transformar os sinais diferenciais do *HyperTransport* em sinais mais simples. Descobriu-se com isto que os sinais físicos do *HyperTransport* são semelhantes aos do LVDS, mas com um offset de *common-mode* diferente. O *output* do conversor é ligado ao FPGA e este é configurado para receber sinais de alta velocidade.

O circuito que foi construído foi soldado directamente no bus *northbridge-southbridge* da Xbox.

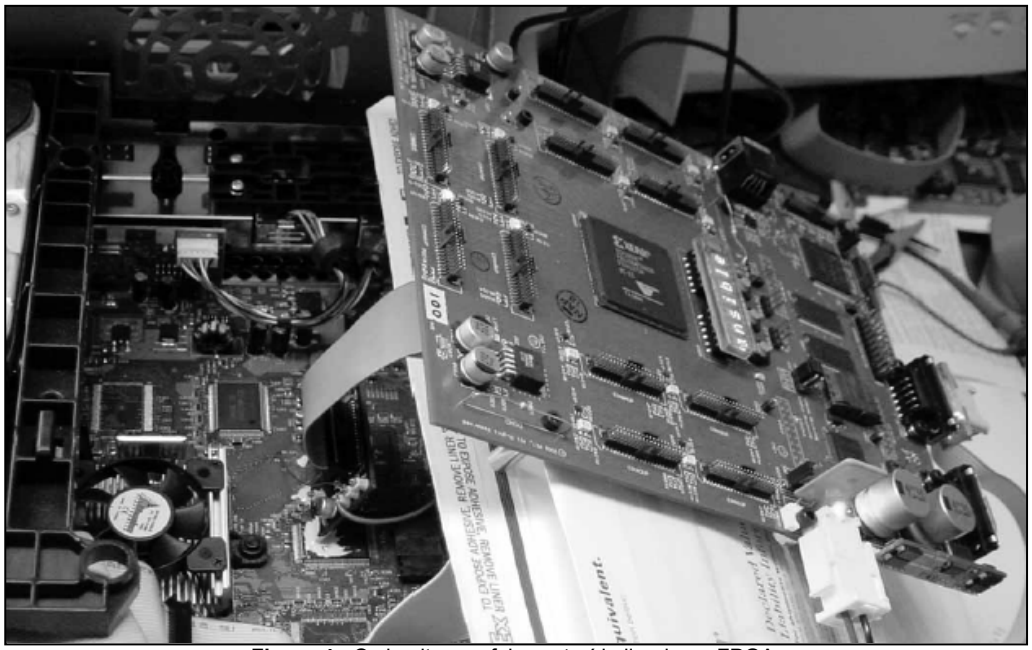

Figura 4 - O circuito que foi construído ligado ao FPGA.

A polaridade dos sinais do bus do *HyperTransport* foi determinada analisando o estado latente dos fios, assumindo que este estado tinha o valor 0x00. Estes sinais tinham os pares negativo e positivo trocados relativamente ao circuito que tinha sido construído para analisar os sinais. Os sinais com a polaridade invertida foram depois restaurados ao seu valor original quando começou a ser feita a analise dos sinais e quando estes entraram no FPGA.

Um FPGA *Xilinx Virtex* foi usado para realizar a análise da actividade do bus *HyperTransport*. Tornou-se uma tarefa complicada fazer com que o FPGA fosse capaz de lidar com os *data rates* de 200 MHz DDR com níveis de distorção baixos. No entanto, a manipulação cuidadosa dos registos de entrada, bem como simulações temporais de *post-layout* a uma temperatura e voltagem nominal e iterações para manualmente provocar atrasos e distorções, eventualmente levou à centragem do sinal do relógio com o sinal de dados nos registos de entrada do FPGA. Em seguida os dados foram desmultiplicados para um bus de banda larga com um data rate de 32 bit a 100 MHz, e escritos para memória usando FIFO, juntamente com uma sequência de contagem que regista em que ciclo, (relativo a um sinal de reset), os dados foram capturados.

O sinal de reset pode ser determinado fazendo uma sondagem de análise perto do bus *HyperTransport* que se comporta como um sinal de reset. Na realidade, é possível que algum dos sinais que não fosse o verdadeiro sinal de reset tivesse sido usado para accionar a captura de análise, mas esse aspecto é irrelevante, pois o sinal escolhido aparentava ter um relacionamento temporal consistente em relação ao bus.

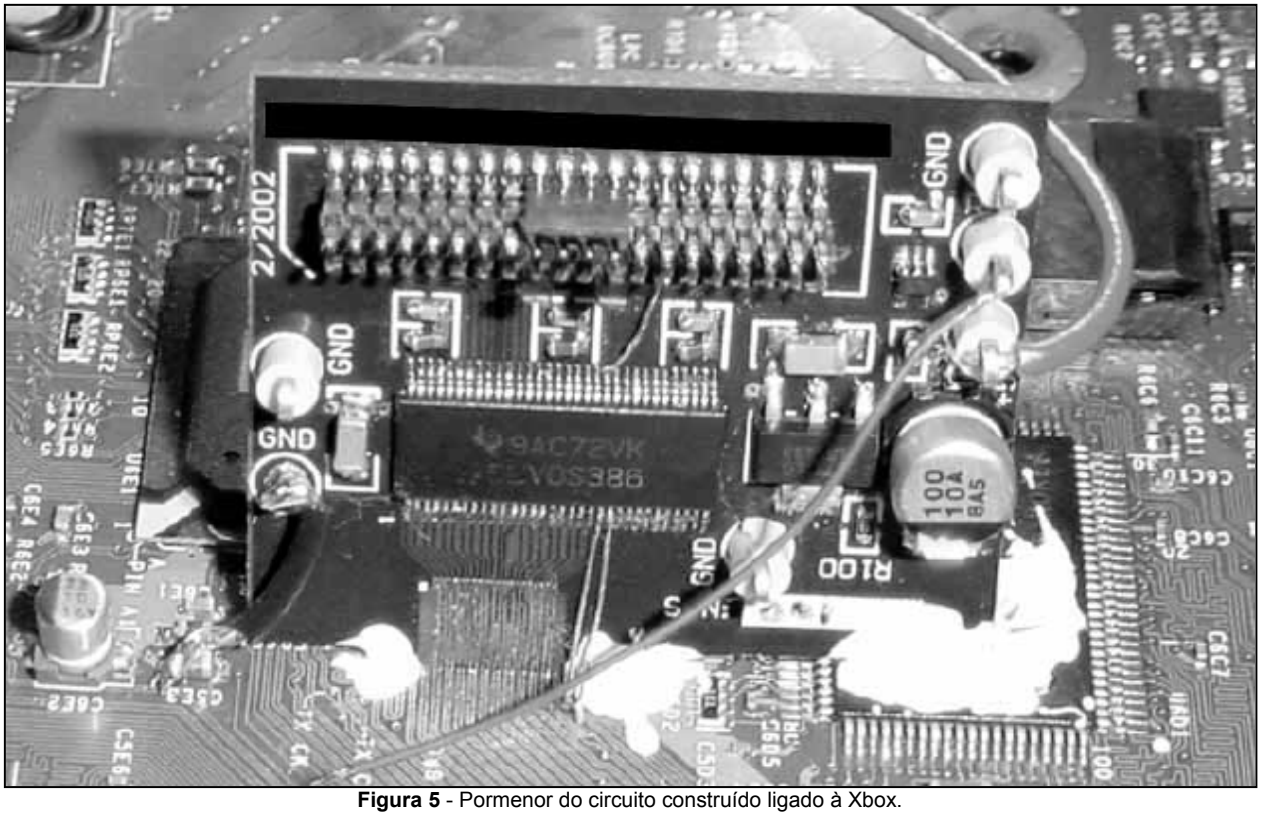

Figura 5 - Pormenor do circuito construído ligado à Xbox.

Depois da análise de dados ter sido capturada pelo FPGA, a ordem dos bits no bus *HyperTransport* relativamente ao circuito construído para analisar o conteúdo do bus pôde ser determinada. Esta determinação pode ser feita através da correlação de valores conhecidos da *FLASH ROM* com valores de dados detectados no bus *HyperTransport*. Pode-se usar uma contagem de "1's" para identificar *candidate patterns* e sequências de dados para correlação manual. Felizmente foram detectados e armazenados logo no início da análise valores sequenciais a partir da *FLASH ROM*: Alguns valores dos endereços mais baixos do *FLASH ROM*, seguidos por outros valores do vector de arranque, que são idênticos ao conteúdo do *FLASH ROM* falso e ao conteúdo da boot ROM escondida.

Os dados capturados pelo FPGA foram depois gravados em ficheiros e processados manualmente. Um exemplo destes dados pode ser visto na figura 6. A sequência dos números foi critica para determinar as ligações com a cache. É feito um *fetch* de blocos de 8 ou 16 words pelo processador, mesmo quando a cache está desactivada. Os dados da análise foram diferenciados entre o *boot code* escondido e os dados do *FLASH ROM*, através da procura da primeira word da análise do *dump* do *FLASH ROM*. Se os dados não forem encontrados no *FLASH ROM* então adivinhava-se que seriam do *boot code* escondido. Devido ao facto do processador arrancar com a cache desligada, os primeiros 24 milhões de ciclos do bus contêm linhas repetidas do código de inicialização da *jam table*. Estes ciclos foram ignorados, pois eram usados para inicialização dos *chipsets*.

As caches foram então ligadas pelo *boot block*, e foram encontrados blocos de instruções de fácil leitura. Estas instruções foram mapeadas para o *boot block* escondido usando a *FLASH ROM* falsa como *template*. O bloco de dados recuperado foi então depurado e, o algoritmo de encriptação usado foi descoberto como sendo o RC-4 de 128 bits. Como a localização da chave de 128 bits dentro do *boot block* secreto era ambígua (uma vez que o circuito usado, não era suficiente para dar este tipo de informação, pois ele apenas faz análises de dados e não de endereços), teve de ser usado um método de força bruta de procura para ajudar a isolar a chave. Foi usada uma chave de 16 bytes com o motor de descodificação RC-4 que serviu de input e foi usado um histograma dos dados de saída para determinar se a chave já tinha sido encontrada. Esta informação ajudou a resolver algumas ambiguidades, pois, assim tornou-se mais fácil ver onde e como ficavam os dados dentro do *boot block* secreto.

Posto isto, é possível encriptar uma nova ROM para a Xbox, e estudar mais aprofundadamente a estrutura do *bootloader* e do *kernel*. Dado o algoritmo RC-4, a chave de 128 bits e o número mágico usado no fim do segmento descodificado, é possível executar código feito por outros na Xbox.

```
00000097 : 664A1D55 ::: E : 000000C6
00000D5C : 05F108F6 ::: F : 01000000
00000DE0 : 2A1A2841 ::: 1 : CC003000
00000E5D : B6FE7F68 ::: E : A0552C01
00000EDA : 5932C662 ::: 1 : 000000FD
00000F57 : F9FBA4C1 ::: E : C7C94000
00000FD4 : F7F9B6AE ::: 1 : 000000C6
00001051 : 73376133 ::: E : 9EC49400
000010CE : FD0127AD ::: 1 : 000000D6
0000114B : 34E8FD29 ::: E : C7C94000
00001245 : 1814A022 ::: 1 : 000000C6
000012C2 : 38EBD672 ::: E : C7C94000
00022526 : C6C0847E ::: 1 : 000000C6
00022527 : A26216BB ::: E : C7C94000
00022528 : 99DA5F80 ::: E : 000000C6
00022529 : 453862E3 ::: 1 : C7C94000
000226D5 : B6DF18C0 ::: E : 000000C6
000226D6 : DA562768 ::: 1 : C7C94000
000226D7 : 0F1D66E3 ::: E : 000000C6
000226D8 : DDC59B59 ::: 1 : 8D42CBCD
```

Figura 6 - Um exemplo do formato dos dados capturados pelo FPGA. O formato dos dados é: "sequência : dados ::: alinhamento : dados desalinhados".

4. Investigações Futuras

Recentemente descobriu-se que a inicialização do hardware da Xbox contêm uma falha relevante. O primeiro passo no processo de arranque da Xbox consiste em fazer um carregamento das *jam tables*, que configuram os *chipsets* da consola. Este processo de inicialização envolve uma sequência de escritas para diferentes registos de memória. Como resultado, o procedimento de inicialização é implementado usando um simples intérprete *bytecode* que lê os comandos e data de inicialização a partir da *FLASH ROM*. Estes comandos do *bytecode*, que são guardados como *plaintext*, podem ser manipulados para provocar que o procedimento de inicialização termine antes que a rotina de verificação da descodificação do *kernel* seja executada, de modo a ser executado código inseguro directamente *da FLASH ROM*. Esta falha poderia ser facilmente colmatada verificando o conteúdo da *jam table* antes da execução do *bytecode* com uma função de hash que executa só num sentido, ou então, programando explicitamente a inicialização das funções dentro

do *boot block* seguro. No entanto ambas estas soluções iriam necessitar que um *boot block* seguro aumentasse significativamente do seu tamanho de 512 bytes. Além disso nenhuma destas soluções permite alterações relativamente fáceis de realizar no procedimento de inicialização caso seja encontrado um *bug*.

Perceber o protocolo de arranque é apenas o primeiro passo para perceber o funcionamento completo da Xbox. É agora possível investigar o *kernel* e o *bootloader* com maior detalhe. Determinou-se que o *kernel* está igualmente encriptado com RC-4 usando uma chave de 128 bits, e também usando compressão LZX, um esquema usado pela Microsoft. A estrutura e função do *kernel* está ainda sobre investigação.

Um importante tema a ser abordado está relacionado com a privacidade que o utilizador da Xbox tem quando realiza tarefas on-line. Sabe-se que a informação tal como o número de série da consola é armazenado digitalmente e poderá provavelmente ser acedido através do kernel. Devido à encriptação usada na protecção da Xbox, a natureza da informação que é enviada para a Microsoft nos servidores de jogos on-line é desconhecida. Consequentemente, investigações futuras concentram-se em determinar que tipo de informação a Xbox revela acerca da identidade do utilizador e seus hábitos pessoais.

5 Conclusões a Retirar

Uma conclusão que se pode retirar deste artigo é que o uso de um único bus de alto desempenho não é uma medida de segurança suficiente. A existência de serviços protocolares rápidos e baratos e de FPGAs de grande performance permitem criar serviços que possam fazer um *trace* ao bus. Contudo, encriptar um bus provoca um certo número de problemas. Uma codificação segura de um bus de alto desempenho provoca latência, consumo de energia, e quebra de fiabilidade. O consumo de energia aumenta porque o factor de actividade do bus chega aos 100% caso o sistema de encriptação seja bom. Caso seja o caso, a energia consumida pelo bus iria crescer em magnitude, pois o factor de actividade observado no bus estava abaixo dos 10%. A fiabilidade é afectada, pois qualquer pequeno erro pode corromper grandes blocos de dados.

Uma solução para este problema seria o de simplesmente não confiar em qualquer bus do sistema. Neste caso, o *boot block* escondido pode usar um protocolo digital como é o caso do *Authenticode*, usando chaves algorítmicas públicas. Depois, toda a segurança do sistema fica dependente do secretismo da chave privada, e na capacidade da chave algorítmica pública. De forma a prevenir que os funcionários distribuam chaves privadas, um sistema semelhante ao BBN *SignAssure* (http://www.is.bbn.com/projects/darpa-chats-cms/index.html) poderia ser utilizado para gerir a chave, para que ninguém pudesse ter conhecimento da chave privada. A principal desvantagem deste método é o de se ter de ter uma área extra de silício para se guardar um *boot block* escondido, já que será quase impossível codificar um algoritmo de encriptação de uma chave pública (mais o armazenamento da chave e inicialização do hardware) em apenas 512 bytes. A sugestão adiantada também não impossibilita que alguém faça *eavesdropping* e que obtenha o código do sistema operativo, embora impossibilite qualquer tentativa de correr código original. O mecanismo da chave pública poderia ser no entanto contornado, por um mecanismo que ataque o bus da memória principal e que faça um *patch plaintext* na memória principal. Esta abordagem é possível embora difícil. A dificuldade de fazer um *patch* à memoria principal pode ser aumentada implementando um *hash* periódico e fazendo uma verificação do código nas zonas criticas da memória.

A exploração do *buffer overrun* põe também à descoberta pontos fracos que estão sempre presentes, independentemente do secretismo do protocolo de arranque. Um eventual *hacker* que explore um bus sem segurança pode obter o código de desencriptação do kernel e analisá-lo em busca de pontos fracos. No entanto, qualquer máquina que use *guarded pointers* é muito mais difícil de atacar usando *buffer overruns.*

É um erro assumir que um segredo, distribuído juntamente com a informação que guarda, nunca será revelado. Por exemplo, a Sega Dreamcast usa o formato de software GD-ROM, mas a *drive* consegue ler CD-ROMs. A descoberta de uma "fuga" no sistema operativo da Dreamcast permite que ficheiros executáveis possam ser corridos directamente de um CD-ROM comum. A descoberta desta "fuga" no sistema operativo que permitia a execução desse tipo de ficheiros anulou por completo a barreira apresentada pelo formato GD-ROM. Outros sistemas que se baseiam em segredos supostamente bem guardados,

como são o caso dos *smartcards* usados em serviços como a Pay-TV e telemóveis, têm-se revelado surpreendentemente vulneráveis.

A criação de uma consola com muita segurança dificulta o seu *debugging* e produção. Por exemplo, a parte de trás da motherboard da Xbox está cheia de pontos teste que incluem pontos teste para cada *pin* da FLASH ROM. Estes pontos foram instalados originalmente com o intuito de se fazerem rapidamente testes para se verificarem eventuais falhas durante a fase de produção. No entanto estes pontos teste podem ser utilizados para reprogramar a motherboard da Xbox com outro código qualquer. Este método seria rápido e sem custos. A conclusão a retirar é a de que mesmo que um fabricante esteja muito confiante acerca do modelo produzido e dos seus protocolos de segurança, deve salvaguardar a possibilidade de que eventualmente toda essa segurança será quebrada um dia. Para evitar que as estruturas de teste fossem usadas para outro fim, bastaria fazer um *spray-on*. Claro que esta opção iria dificultar em muito a tarefa de reparação da consola, mas são opções que sem de ter tomadas pelo fabricante. Uma alternativa mais extrema seria a de desenvolver formatos de media patenteados, limitando assim um impacto prático de qualquer ataque feito à consola. O formato ao ser patenteado daria protecção contra qualquer tipo de uso inapropriado, sem haver necessidade de se manter em segredo.

Perguntas e Respostas

1. Quais são os problemas associados à encriptação dos busses da Xbox?

Um bus geralmente não é encriptado, pois considera-se que é seguro devido às suas altas velocidades. No entanto, se se tiver acesso a algum hardware específico, torna-se possível efectuar um *trace* ao bus. No entanto, a encriptação de um bus é problemática, na medida que provoca problemas de fiabilidade, consumo de energia e latência.

O consumo de energia aumenta porque o factor de actividade do bus chega aos 100% caso o sistema de encriptação seja bom. Neste caso, a energia consumida pelo bus iria crescer em magnitude, pois o factor de actividade observado no bus northbrigde-

southbridge estava abaixo dos 10%. A fiabilidade é afectada, pois qualquer pequeno erro num bit, mesmo durante um ciclo *idle* pode corromper grandes blocos de dados. Com uma *stream* codificada, a zona de dados corrompida aumenta até que a *stream* seja resincronizada.

2. Qual é o método que os fabricantes da Xbox utilizaram para camuflar o boot block do sistema? Descreva as funções que este desempenha no arranque da Xbox.

A Xbox possui dois *boot blocks* no seu sistema, sendo um deles falso, encontrando-se este na memória de leitura (ROM) externa (ver Figura 1). O *boot block* falso existe como forma de tentar evitar possíveis alterações ao funcionamento normal da consola (dar outras funcionalidades à Xbox, como por exemplo instalar o Linux ). Assim, este boot block falso serve de "isco", pelo facto de conter algum código de inicialização. O verdadeiro *boot block* tem 512 bytes e encontra-se escondido integrado no sistema *southbridge ASIC.*

As funções realizadas pelo *boot block* escondido no arranque da Xbox incluem os seguintes procedimentos:

- Carregar as *jam tables*;
- Ligar as caches do processador;
- Descodificar o *bootloader* do *kernel*, contido na *FLASH ROM;*
- Confirmar se a descodificação foi bem sucedida;
- Passar o controlo para o *bootloader* do *kernel* descodificado.

Estas operações são realizadas por esta ordem pelo *boot block*.

Agradecimentos

Os autores agradecem aos pais e irmãos que muito ajudaram e a todos os colegas que os incentivaram e ajudaram na pesquisa e elaboração do artigo.

Referencias

- IBM, *IBM 4758 PCI Cryptographic Coprocessor*, http://www.ibm.com/security/cryptocards/
- Hacking The Xbox, http://hackingthexbox.com/
- HyperTransport Consortium, *HyperTransport I/O Link Specification, Version 1.03*, http://www.hypertransport.org
- nVidia Corporation, *nForce MCP Product Overview, 06.01v1*, http://www.nvidia.com
- Microsoft Developer Network, *Introduction to Code Signing*, http://msdn.microsoft.com/workshop/security/authcode/intro authenticode.asp
- The Xbox Hacker, http://www.xboxhacker.net
- The Xbox Linux Project, http://xbox-linux.sourceforge.net/
- The Xbox Scene, http://www.xbox-scene.com/