Sistemas Embarcados - Projeto Final (TP2)

O projeto final da disciplina Sistemas Embarcados consiste na prototipação de um SoC composto por um processador, periféricos e um bloco de criptografia 3DES, além do sistema operacional UCX/OS e um driver que suporte os modos de operação ECB, CBC e CTR para implementação de criptografia de fluxo.

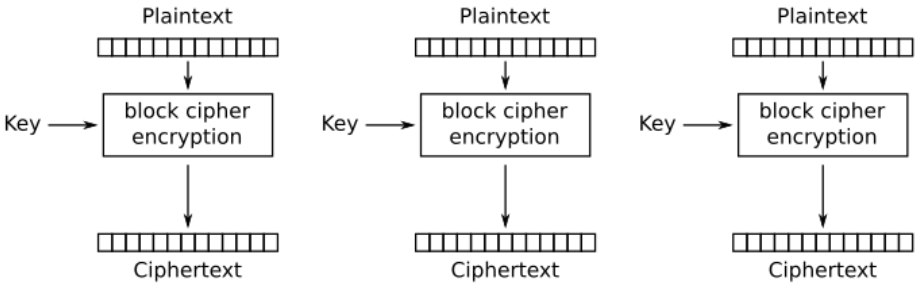
Descrição:

Uma plataforma para o desenvolvimento de sistemas embarcados baseada no SoC HF-RISC e no RTOS UCX/OS foi apresentada em aula. Esse SoC possui diferentes versões de processador, *wrappers* para simulação e prototipação e periféricos. O objetivo é que nesse projeto final, o seu grupo realize a integração de um bloco criptográfico 3DES descrito como hardware ao SoC, e implemente um driver utilizando o UCX/OS que permita que a aplicação embarcada possa realizar o processo de criptografia e descriptografia de mensagens de tamanho arbitrário.

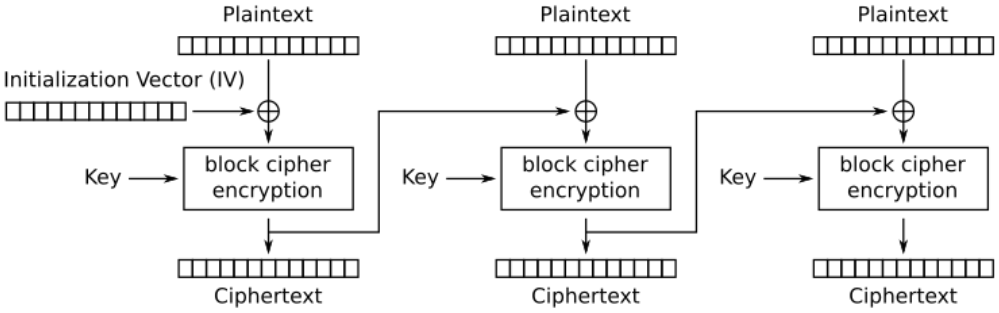
Para realização do processo de criptografia de informação de mensagens com tamanho arbitrário, a utilização de um modo de operação[1] de um *block cipher* é necessário, uma vez que a maioria dos algoritmos de criptografia operam com blocos de tamanho fixo e utilizam um sistema de chave simétrica. Os modos a serem implementados consistem no ECB, CBC e CTR. O Modo EBC é bastante simples, uma vez que os blocos são processados um a um usando criptografia de bloco sem cuidados adicionais. O modo CBC implementa um sistema de criptografia mais forte, uma vez que o processamento de blocos consecutivos dependem do resultado anterior. O modo CTR utiliza um contador juntamente com um vetor de inicialização, o que permite paralelizar mais facilmente o processamento. Uma característica interessante do modo CTR, é que apenas é necessário o processo de criptografia do algoritmo de bloco de tamanho fixo para implementação dos processos de criptografia

[1] https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation

e descriptografia de mensagens de tamanho arbitrário. Além disso, cada bloco pode ser processado de forma independente (semelhante ao modo ECB (*Electronic Codebook*)).

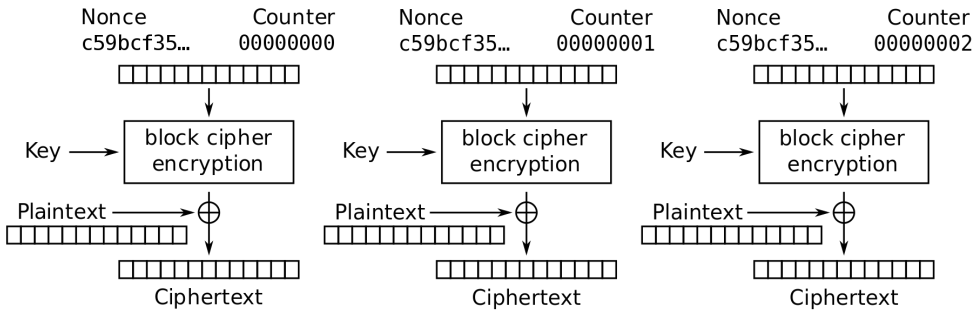


Electronic Codebook (ECB) mode encryption



Cipher Block Chaining (CBC) mode encryption

O bloco de criptografia exemplo (XTEA) deverá ser substituído pelo bloco 3DES fornecido em conjunto com o enunciado. Além disso, será necessário definir no *top level* (*testbench*) uma instância de um módulo (por meio de *port map*) a ser desenvolvido, que irá definir uma região de endereços e registradores para interface com o driver em software, além da instanciação dos componentes e a lógica de cola do bloco 3DES com o SoC. Na descrição de referência (que pode ser simulada pelo script *hf-risc/sim/rv32e_basic_xtea*) e que inclui o testbench *hf-risc/riscv/sim/hf-riscv_basic_soc_xtea_tb.vhd* como *top level* isso é feito diretamente, o que não deve acontecer na sua implementação.



Counter (CTR) mode encryption

Para o entendimento de como controlar o bloco criptográfico, podem ser utilizadas as implementações de referência dos exemplos *xtea, xtea2, xtea3, xtea_hw*[2] e *xtea_hw_vec*, disponibilizadas com os fontes do repositório HF-RISC. Nos exemplos *xtea_hw* e *xtea_hw_vec* são apresentadas implementações de comandos de controle para acesso ao hardware e exemplos de uso de um bloco XTEA integrado ao hardware. No exemplo *xtea2*[3], são realizados os processos de criptografia e descriptografia de uma mensagem utilizando diferentes modos de operação (CBC (*Cipher Block Chaining*) e CTR (*Counter*)).

O seu driver deverá utilizar a API de device drivers genérica do UCX/OS, a qual implementa operações para inicialização (*dev_init()*), abertura e fechamento (*dev_open(), dev_close()*) e operações de escrita e leitura (*dev_write(), dev_read()*). Operações de escrita são utilizadas para o envio de dados ao driver, que deverá realizar o processo de criptografia ou descriptografia de fluxo. O processo envolve usar um modo de operação implementado em software e acessar o módulo criptográfico 3DES no hardware para criptografia de bloco. O driver deve armazenar o resultado do último processamento após um comando de escrita, gerenciando o processo de alocação de memória. Em uma operação de leitura, o resultado deve ser copiado para um buffer do usuário e a memória alocada previamente deve ser liberada. A exclusão mútua para acesso ao driver será implementada por meio das operações de abertura e fechamento de um dispositivo, sendo que operações de escrita ou leitura só podem ser realizadas se uma tarefa conseguir abrir o dispositivo.

[2]Código fonte *hf-risc/software/xtea_hw.c*
[3]Código fonte *hf-risc/software/app/xtea2.c*

São fornecidos os seguintes recursos a serem utilizados para o desenvolvimento do trabalho:

- Repositório do SoC HF-RISC (*https://github.com/sjohann81/hf-risc*);
- Repositório do RTOS UCX-OS (*https://github.com/sjohann81/ucx-os*);
- Implementação e documentação do módulo 3DES;

Recomenda-se que o grupo inicialmente familiarize-se com o material disponibilizado, executando a aplicação *xtea_hw* em uma plataforma simples (*hf-risc/sim/rv32e_basic_xtea*, repositório HF-RISC). Como apresentado em aula, é possível simular a plataforma utilizando diferentes ferramentas, como ISE, ModelSim ou GHDL. Após, é sugerido que o grupo realize a configuração e execução do UCX-OS na plataforma, explorando aplicações como *hello_p* e exemplos envolvendo drivers de dispositivos. Um exemplo simples de driver que pode ser utilizado como referência para implementação do trabalho está disponível em *ucx-os/app/driver*. Outros exemplos para drivers mais elaborados estão no diretório *ucx-os/drivers*. Nesse diretório estão também disponíveis os recursos que implementam as interfaces e drivers para diversos dispositivos.

Entrega:

O trabalho deve ser realizado em duplas ou individualmente e entregue via Moodle. Envie os fontes implementados (software e hardware), juntamente com um relatório explicando a organização do projeto, a estratégia utilizada para solucionar o problema e a validação da integração do bloco criptográfico por meio de simulação. Na validação, deve ser apresentado o processamento de mensagens de tamanho variado, utilizando criptografia de fluxo nos três modos. É importante ilustrar os processos de criptografia e descriptografia realizados por aplicações exemplo, as quais acessam a API do driver desenvolvido. O trabalho será apresentado em aula e para isso é necessário que o grupo se organize para uma apresentação de aproximadamente 10 minutos.